1992

The design and accuracy study of 2-D DCT/IDCT processors

Maurice D. Caldwell
Lehigh University

AUTHOR:

Caldwell, Maurice D.

TITLE:

The Design and Accuracy Study of 2-D DCT/IDCT Processors

DATE: May 31, 1992

The Design and Accuracy Study of 2-D DCT/IDCT Processors

by

Maurice D. Caldwell

A Thesis

Presented to the Graduate Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Electrical Engineering

Lehigh University

May 1992

This thesis is accepted and approved in partial fulfillment of the requirements for the Master of Science in Electrical Engineering.

May 6, 1992
Date

Advisor in Charge

EECS Department Chairperson

Acknowledgements

I would like to thank Dr. Weiping Li for his unyielding support and encouragement on this project, for believing in my abilities, and for inspiring me to strive for the top. I thank my mother, Jacquie Caldwell, my brother, John, and my sister, Simone, for their guidance and support. I thank Michelle Ware and Sam Boynton for their unwavering support throughout my Masters program. Finally, I thank my roommates Mike Gosse, Paul Kraft, Kevin Williams, Chuck Winstead, and Byron Bratcher for those late night sessions.

Table of Contents

List of Tables

List of Figures

ABSTRACT

Recently, IEEE, in conjunction with CCITT, developed a set of standard specifications, and accuracy testing procedure, for 8x8 Inverse Discrete Cosine Transform implementations used in video and image processing. In addition, a new fast algorithm for computing the Discrete Cosine Transform and its Inverse has been derived. In this study, it is shown that a previously designed 2-dimensional DCT/IDCT chip, which is based on the new algorithm, fails to satisfy the standard specifications. As a result, two distinct 2-D DCT/IDCT processor designs are presented which significantly satisfy the standard specifications. Included in this study are: an introduction of the DCT and of the new fast algorithm, a thorough analysis of the two designs, results from the accuracy testing procedure, and a proposal for which design should be implemented.

CHAPTER 1: INTRODUCTION

The Discrete Cosine Transform (DCT) is considered to be the most widely used data compression tool for image processing. N. Ahmed, T. Natarajan, and K. R. Rao first introduced the DCT algorithm in 1973 [1]. It was computed using the Fast Fourier Transform (FFT). However, the DCT computation has since evolved in a manner similar to evolution of the Fourier Transform due to the FFT and other fast algorithms. Reduction of computational redundancies in the DCT algorithm has led to wide-spread use of DCT VLSI micro chips in digital signal processing. Specific applications include: High definition television (HDTV), teleconferencing, videophones, videotext, filtering, multispectral scanner data, speech coding, image coding, etc [2]-[11].

The DCT is a real, separable, and orthogonal transform that compares well with the statistically optimal Karhunen-Loeve Transform (KLT) which serves as a reference for measuring transforms. Separable means that one dimensional DCT's in series can be used to implement multidimensional DCT's. And, orthogonal means that a signal can be separated (transformed) into components which, as a whole, represent the original signal information. An optimal transform (KLT) for data compression has the following properties: 1) it completely decorrelates the signal in the transform domain: 2) it contains the most variance (energy) in the fewest

number of transform coefficients: and 3) it minimizes the total representation entropy of the sequence. The KLT can be thought of as a transform that maps a highly correlated N point data sequence into N uncorrelated coefficients that represent the same signal. The KLT is rendered an ideal transform since actual implementation depends on the ability to predetermine the basis vectors. The DCT, however, is able to closely approximate the KLT basis vectors. When applied to data that is highly correlated, the DCT will concentrate higher energies in the lower order transform coefficients [1], [2].

A two-dimensional DCT as applied to image processing operates on NxN square blocks of pixels, where a pixel is a colored dot on a television screen. A standard TV screen contains 525 lines of pixels and HDTV contains 1,125 lines. The separability property of the DCT suggests that two one-dimensional DCT's in series can yield a two-dimensional DCT result. However, the NxN DCT result, after the first one-dimensional DCT, must be transposed before entering the second one-dimensional DCT. Transpose means that rows become columns. The two-dimensional DCT will redistribute higher energies into the upper left corner (lower order transform coefficients) of the NxN block.

In processing units, pixel values are represented by eight bits and the DCT results are represented by 12 bits. As mentioned, the DCT is a popular data compression tool in

image processing. However, the results of a two-dimensional DCT on an NxN pixel block yields NxN DCT coefficients. Thus, the DCT alone does not compress the data. Actually, the DCT is used in conjunction with coding algorithms and/or thresholding algorithms to achieve data compression. Coding and thresholding algorithms reduce the number of bits required to represent an NxN pixel block by: representing higher transform coefficients (lower energies) with less than 12 bits, and discarding higher transform coefficients near zero, respectively.

In the past, DCT micro chip manufacturers would find themselves trying to determine a fine median between chip size and speed, and computational accuracy and image quality. Here, it was left up to the designer to balance this trade-off. Thus, there existed chips with different accuracy capabilities. Specific applications exists, however, that call for computational accuracy to be within a tolerable bound for image quality degradation to be negligible. Recently, IEEE has established "IEEE Standard Specifications for the Implementations of 8x8 Inverse Discrete Cosine Transform" [11]. This serves as a marker for designers to meet so that image quality is maintained regardless of whose chip is implemented in an application.

The material presented in this thesis is organized into seven chapters. Chapter 2 gives an overview of the original DCT algorithm. Chapter 3 introduces the new fast DCT

algorithm maps out the computations required by hardware. Chapter 4 discusses, thoroughly, the IEEE accuracy testing and standard specifications procedure. In Chapter 5, two distinct hardware implementations of a two-dimensional DCT/IDCT processor are presented. The results of the two designs against the standards is covered in Chapter 6. In addition, results of the "original design" are introduced here. Finally, Chapter 7 discusses the advantages between the designs.

CHAPTER 2: OVERVIEW OF THE DCT

The practical limitations regarding the calculation of the KLT, due to its inability to predetermine its basis vectors, led to the DCT algorithm in the early 70's by Ahmed, Natarajan, and Rao [1]. They determined that the basis set for the DCT very closely approximates the eigenvectors of the Toeplitz matrix.

DCT Basis Set:

$$\left[\frac{1}{\sqrt{2}}, \cos\left(\frac{(2n+1)k\pi}{2N}\right)\right]$$

where k = 1, 2, ..., N-1 and n = 0, 1, ..., N-1.

Toeplitz Matrix:

$$\psi = \begin{bmatrix} 1 & \rho & \rho^2 & . & . & . & \rho^{N-1} \\ \rho & 1 & \rho & . & . & . & \rho^{N-2} \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ \rho^{N-1} & \rho^{N-2} & . & . & . & . & 1 \end{bmatrix}$$

where ρ is the adjacent correlation factor, $0 < \rho < 1$. The DCT best approximates the KLT when the input data is highly correlated ($\rho \sim 0.9$). The DCT and the IDCT of an N point sequence are defined in Equations (1) and (2), respectively.

$$X(0) = \sqrt{\frac{1}{N}} \sum_{n=0}^{N-1} x(n) \qquad (1a)$$

$$X(k) = \sqrt{\frac{2}{N}} \sum_{n=0}^{N-1} x(n) \cos\left(\frac{\pi(2n+1)k}{2N}\right) \qquad (1b)$$

where k = 1, 2, .., N-1.

And,

$$x(n) = \sqrt{\frac{1}{N}}\, X(0) + \sqrt{\frac{2}{N}} \sum_{k=1}^{N-1} X(k) \cos\left(\frac{\pi(2n+1)k}{2N}\right) \qquad (2)$$

where n = 0, 1, ..., N-1.

The original derivation shows that a 2N point Discrete Fourier Transform (DFT) can be used to compute the DCT of an N point real signal. Since its introduction, many fast algorithms for computing the DCT have been developed. The more popular fast algorithm, introduced by Narasimha and Peterson, shows that the same result can be obtained using an N point DFT with the input sequence rearranged [4]. Fast algorithms which convert the DCT computation to that of circular convolution have also been developed. Circular convolution algorithms are suitable for distributed arithmetic configurations in hardware implementation.

CHAPTER 3: THE NEW DCT ALGORITHM

A new fast algorithm for computing the DCT and its Inverse (IDCT) has been derived by Dr. Weiping Li [12]. This algorithm converts the DCT computation to that of skew-circular (SC) convolution. The DCT computation, now in terms of SC convolution, enables distributed arithmetic to be implemented when designing a VLSI chip. Distributed arithmetic is the hardware design technique that stores in memory precalculated combinations of the fixed convolution matrix coefficients. This technique greatly reduces hardware components since the number of calculations required by the processor is reduced.

The new fast algorithm for the computation of an N point DCT and IDCT, as summarized in [12], is given below. See [12] for derivation of the new algorithm.

Summary of DCT Computation

1) Re-order the N point input sequence {x(n)} to obtain new sequence $\{x(n_i)\}$ for i = 0, 1, ..., N - 1 according to

$$n_i = \begin{cases} \dfrac{3^i \bmod(4N) - 1}{2} & \text{if } \dfrac{3_i \bmod(4N) - 1}{2} < N \\ 2N - 1 - \dfrac{3^i \bmod(4N) - 1}{2} & \text{if } \dfrac{3^i \bmod(4N) - 1}{2} \geq N. \end{cases} \tag{3}$$

2) Obtain sequence of length N/2 according to subtractions

$$y(i) = x(n_i) - x(n_{(N/2)+i}) \tag{4}$$

for i = 0, 1, ..., (N/2) - 1.

3) Obtain $X'(k_j)$, for j = 0, 1, ..., N/2 - 1, by computing an N/2 point SC convolution of the sequence {y(i)} and a constant sequence such that

$$X'(k_j) = \sum_{i=0}^{(N/2)-1} y(i) \left[\sqrt{\frac{2}{N}} \cos\left(\frac{2\pi}{4N} 3^{i+j}\right)\right]. \tag{5}$$

4) Map j to k_j, for j = 0, 1, ..., (N/2) - 1, according to

$$k_j = \begin{cases} \dfrac{3^j \bmod(4N) - 1}{2} & \text{if } \dfrac{3^j \bmod(4N) - 1}{2} < N \\ 2N - 1 - \dfrac{3^j \bmod(4N) - 1}{2} & \text{if } \dfrac{3^j \bmod(4N) - 1}{2} \geq N. \end{cases} \tag{6}$$

and, obtain DCT odd index components according to

$$\begin{aligned} X(2k_j + 1) &= X'(k_j) \quad \text{if } k_j < \frac{N}{2} \\ X(2(N - 1 - k_j) + 1) &= X'(k_j) \quad \text{if } k_j \geq \frac{N}{2}. \end{aligned} \tag{7}$$

5) Obtain new N/2 point sequence $\{x_1(n)\}$ according to

$$x_1(n) = x(n) + x(N - 1 - n) \tag{8}$$

where n = 0, 1, 2, ..., (N/2) - 1.

6) Repeat Steps 1 through 4 for sequence $\{x_1(n)\}$, with N/2 replacing N, to obtain N/4 DCT components according to {X(4k + 2)}.

7) Repeat Step 5 on sequence $\{x_1(n)\}$, with N/2 replacing N, to obtain new N/4 point sequence $\{x_2(n)\}$. Repeat Steps 1 through 4, N/4 replacing N, to obtain N/8 DCT components according to {X(8k + 4)}.

8) Iterate these steps until a 1-point sequence results from performing Step 5. This is the DCT component X(0) given by

$$X(0) = \sqrt{\frac{1}{N}} \sum_{n=0}^{N-1} x(n) . \tag{9}$$

Summary of IDCT Computation

The IDCT computation is very similar to that of the DCT. The IDCT can be summarized according to

$$\begin{aligned} x(n) &= S_1(n) + S_2(n) \\ x(N-1-n) &= S_1(n) - S_2(n) \end{aligned} \tag{10}$$

where n = 0, 1, ..., (N/2) - 1,

and,

$$S_1(n) = \sum_{k=0}^{(N/2)-1} X(2k) \cos\left(\frac{\pi(2n+1)k}{N}\right) \tag{11}$$

$$S_2(n) = \sum_{k=0}^{(N/2)-1} X(2k+1) \cos\left(\frac{\pi(2n+1)(2k+1)}{2N}\right)$$

where n = 0, 1, ..., (N/2) - 1.

Results of N = 8 Case

Applying the new algorithm, for the N = 8 case, shows that the DCT and IDCT coefficient computations are broken up into four separate computational building blocks: 1) an 8x4 unit that yields the odd indices {1, 3, 5, 7}: 2) a 4x2 unit that yields the even indices {2, 6}: 3) a 2x1 unit that yields index {4}: and 4) a scaler unit that yields the

dc component, index {0}. The following computations are generated by the algorithm [12], [13].

DCT 8x4 Unit:

Input

$$\{x(0), x(1), x(2), x(3), x(4), x(5), x(6), x(7)\} \tag{12}$$

⇓

Input Mapping

$$\{x(0), x(1), x(4), x(2), x(7), x(6), x(3), x(5)\} \tag{13}$$

⇓

Subtractions

$$\begin{aligned}\{x(0)-x(7), x(1)-x(6), x(4)-x(3), x(2)-x(5)\} \\ = \{y(0), y(1), y(2), y(3)\}\end{aligned} \tag{14}$$

⇓

Four Point Skew-Circular Convolution

$$\begin{bmatrix} X'(k_0) \\ X'(k_1) \\ X'(k_2) \\ X'(k_3) \end{bmatrix} = \begin{bmatrix} C(0) & C(1) & C(2) & C(3) \\ C(1) & C(2) & C(3) & -C(0) \\ C(2) & C(3) & -C(0) & -C(1) \\ C(3) & -C(0) & -C(1) & -C(3) \end{bmatrix} \begin{bmatrix} y(0) \\ y(1) \\ y(2) \\ y(3) \end{bmatrix} \tag{15}$$

⇓

Convolution Results

$$\{X'(k_0), X'(k_1), X'(k_3), -X'(k_2)\} \tag{16}$$

⇓

Output Mapping

$$= \{X(1), X(3), X(5), X(7)\} \tag{17}$$

where

$$C(m) = \sqrt{\frac{2}{8}} \cos\left(\frac{\pi 3^m}{2 \cdot 8}\right) = \frac{1}{2} \cos\left(\frac{\pi 3^m}{16}\right). \tag{18}$$

DCT 4x2 Unit:

Input

$$\{x(0), x(1), x(2), x(3), x(4), x(5), x(6), x(7)\}$$

$$\Downarrow$$

Input Mapping & Additions

$$\begin{aligned} &\{x(0)+x(7), x(1)+x(6), x(2)+x(5), x(3)+x(4)\} \\ &= \{x_1(0), x_1(1), x_1(2), x_1(3)\} \end{aligned} \tag{19}$$

$$\Downarrow$$

Subtractions

$$\begin{aligned} &\{x_1(0)-x_1(3), x_1(1)-x_1(2)\} \\ &= \{y_1(0), y_1(1)\} \end{aligned} \tag{20}$$

$$\Downarrow$$

Two Point Skew-Circular Convolution

& Output Mapping

$$\begin{bmatrix} X(2) \\ X(6) \end{bmatrix} = \begin{bmatrix} C_1(0) & C_1(1) \\ C_1(1) & -C_1(0) \end{bmatrix} \begin{bmatrix} y_1(0) \\ y_1(1) \end{bmatrix} \tag{21}$$

$$\Downarrow$$

$$\{X(2), X(6)\}$$

where

$$C_1(m) = \sqrt{\frac{2}{8}} \cos\left(\frac{\pi 3^m}{8}\right) = \frac{1}{2} \cos\left(\frac{\pi 3^m}{8}\right). \qquad (22)$$

DCT 2x1 Unit:

Input

$$\{x_1(0), x_1(1), x_1(2), x_1(3)\} \qquad (23)$$

$$\Downarrow$$

Additions

$$\{x_1(0)+x_1(3), x_1(1)+x_1(2)\}$$
$$\{x_2(0), x_2(1)\} \qquad (24)$$

$$\Downarrow$$

Subtraction

$$\{y_2(0) = x_2(0) - x_2(1)\} \qquad (25)$$

$$\Downarrow$$

One-Point Skew-Circular Convolution

& Output

$$X(4) = Y_2(0) * C_2 \qquad (26)$$

where

$$C_2 = \sqrt{\frac{2}{8}} \cos\left(\frac{2\pi}{8}\right) = \frac{1}{2} \cos\left(\frac{\pi}{4}\right). \tag{27}$$

DCT Scaler Unit:

Input

$$\{x_2(0), x_2(1)\} \tag{28}$$

⇓

Addition

$$\{x_3(0) = x_2(0) + x_2(1)\} \tag{29}$$

⇓

Scaler Multiplication

$$X(0) = x_3(0) * C_3 \tag{30}$$

where

$$C_3 = \sqrt{\frac{1}{8}} = C_2. \tag{31}$$

Notice that all of the building blocks, except for the Scaler Unit, contain subtraction computations just before the SC convolution. The building blocks (8x4, 4x2, & 2x1 units) contain the subtraction and SC convolution computations and, thus, should be considered as an entire unit. It is seen that the subtraction computations generate

the matrix multiply sequence {y(i)} used in the SC convolution. The building block for the Scaler Unit, however, only contains a scaler multiplication computation.

Having established the building blocks, the DCT shows that a series of input mappings and additions precede the blocks. This conditioning of the input sequence is called the Preadd Unit and is only associated with the DCT computation.

In the IDCT, the input sequence goes directly to the building blocks, thus bypassing the Preadd Unit. The building blocks generate a new sequence as follows.

IDCT 8x4 Unit:

Input sequence {X(1),X(3),X(5),X(7),0,0,0,0}

generates sequence $\{S_2(0),S_2(1),S_2(2),S_2(3)\}$.

IDCT 4x2 Unit:

Input sequence {X(2),X(6),0,0}

generates sequence $\{S_2'(0),S_2'(1)\}$.

IDCT 2x1 Unit:

Input sequence {X(4),0} generates sequence $\{S_2''\}$.

IDCT Scaler Unit:

Input sequence {X(0)} generates sequence $\{S_1''\}$.

These new sequences are conditioned by a series additions and subtractions to yield the IDCT results. Accordingly, this unit is called Postadd and is only associated with the IDCT computation. The Postadd consists of the following computations.

Postadd:

$$S_1'(0) = S_1'' + S_2''$$

$$S_1'(1) = S_1'' - S_2''$$

$$S_1(0) = S_1'(0) + S_2'(1)$$

$$S_1(1) = S_1'(1) + S_2'(0)$$

$$S_1(2) = S_1'(1) - S_2'(0)$$

$$S_1(3) = S_1'(0) - S_2'(1)$$

$$IDCT \begin{cases} X(0) = S_1(0) + S_2(0) \\ X(1) = S_1(1) + S_2(1) \\ X(2) = S_1(2) + S_2(2) \\ X(3) = S_1(3) + S_2(3) \\ X(4) = S_1(0) - S_2(0) \\ X(5) = S_1(1) - S_2(1) \\ X(6) = S_1(2) - S_2(2) \\ X(7) = S_1(3) - S_2(3) \end{cases}$$

Implementing the new algorithm into a hardware design requires: 1) an 8x4 Unit: 2) a 4x2 Unit: 3) a 2x1 Unit: 4) a Scaler Unit: 5) a Preadd Unit for the DCT computation: 6) a Postadd Unit for the IDCT computation: and 7) a control unit

that will bypass the appropriate post or pre units.

CHAPTER 4: ACCURACY TEST & SPECIFICATIONS

Recently, IEEE, in conjunction with CCITT, developed "IEEE Standard Specifications for the Implementations of 8x8 Inverse Discrete Cosine Transform" [11]. This documents a standard set of specifications, and an accuracy testing procedure for 8x8 IDCT implementations. The motivation behind the standards is to ensure an upper bound of error due to the numerical accuracy limitations of IDCT hardware implementations.

Manufacturers of IDCT chips use different hardware schemes, in which case, the accuracies differ. This difference, however small, can result in slightly different outputs. In specific applications, such as hybrid Differential Pulse Code Modulation (DCT/DPCM) video coding systems, the IDCT results are used to reconstruct pictures, as illustrated in Fig. 1 [3]. The IDCT is used in the reconstruction loops of the encoder and the decoder. The operation of the hybrid system is as follows.

The picture frame is separated into 8x8 pixel blocks. The memory unit in the encoder, and the decoder, are initialized with the pixel values of the first block. This is known as the intra-frame mode (switch position 2), or as "refreshing" the system. Hereafter, the switch is in position 1. As the current video input block W_i arrives at the input to the system, the memory unit outputs the previous block $\hat{W}_{i-1}$ and the difference of the two is

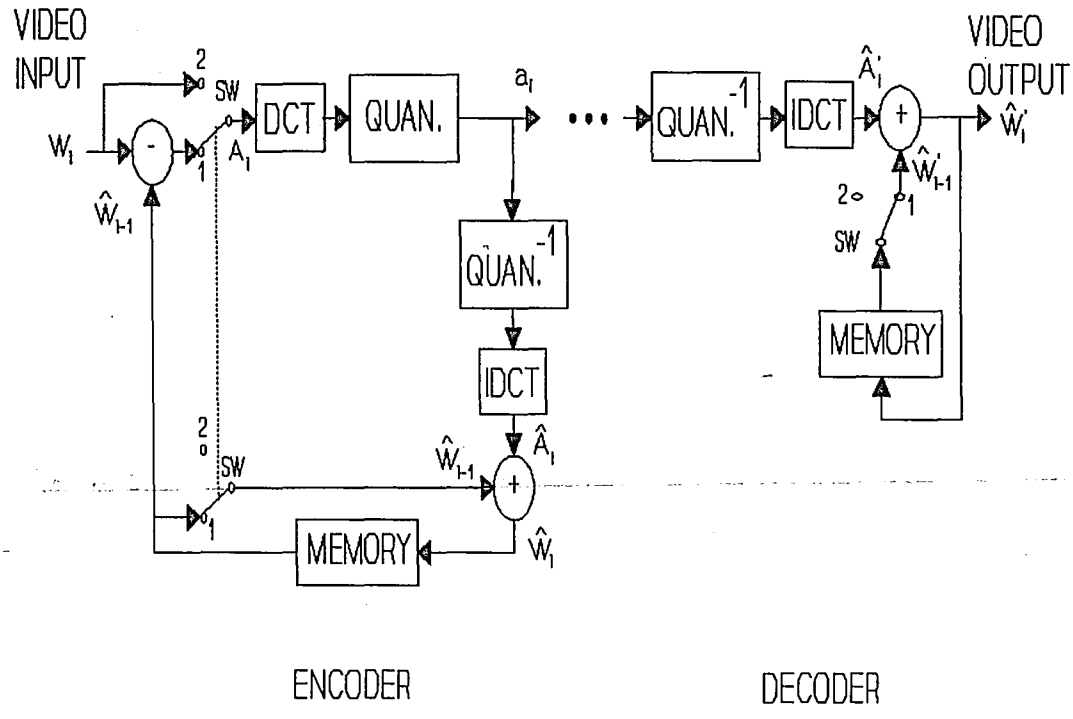

Fig 1: A simplified block diagram of an Encoder-Decoder (CODEC) for a Hybrid DCT/DPCM system.

generated (A_i). This difference (A_i) is then coded, sent through the 8x8 DCT, and quantized to form the result a_i. The result a_i is then transmitted to the decoder. It is used as the input to the reconstruction loop of the encoder and of the decoder. A reconstruction loop contains the inverse quantizer, the 8x8 IDCT, the adder, and the memory unit. In both the encoder, and the decoder, a_i is inversely quantized and sent through the 8x8 IDCT to form $\hat{A}_i$ and $\hat{A}'_i$, respectively. These results are added to their respective previous block memory outputs ($\hat{W}_{i-1}$ and $\hat{W}'_{i-1}$) to form the reconstructed blocks $\hat{W}_i$ and $\hat{W}'_i$, which are then stored in

memory. $\hat{W}'_i$ is the video output. The system is then ready to process the next block.

If the 8x8 IDCT chips in the encoder and the decoder are identical in numerical accuracies, then $\hat{A}_i$ and $\hat{A}'_i$ will be identical. However, if there is a mismatch, then error will accumulate in the memory units. Because only the difference between blocks of adjacent frames are transmitted, this system is very dependant on previous results. If a particular pixel location, for example, incurred a constant error of positive one, due to mismatch, then the accumulated error after n frames would be n. This error is known as mismatch error and can cause severe picture quality degradation in the reconstructed picture.

There are two methods for dealing with mismatch error. The first is re-initialize the system by resetting the memory using the intra-frame mode discussed above. The amount of resetting required is proportional to the error of the system. The other method is to eliminate mismatch error. This implies that the IDCT chips in the encoder and the decoder have to be identical which is very unlikely.

However, IEEE determined a median between the two methods. A set of standards for the numerical accuracy of IDCT chips have been established that, in conjunction with intra-frame refreshing, reduces mismatch error and results in negligible picture quality degradation.

The following is the accuracy testing procedure as

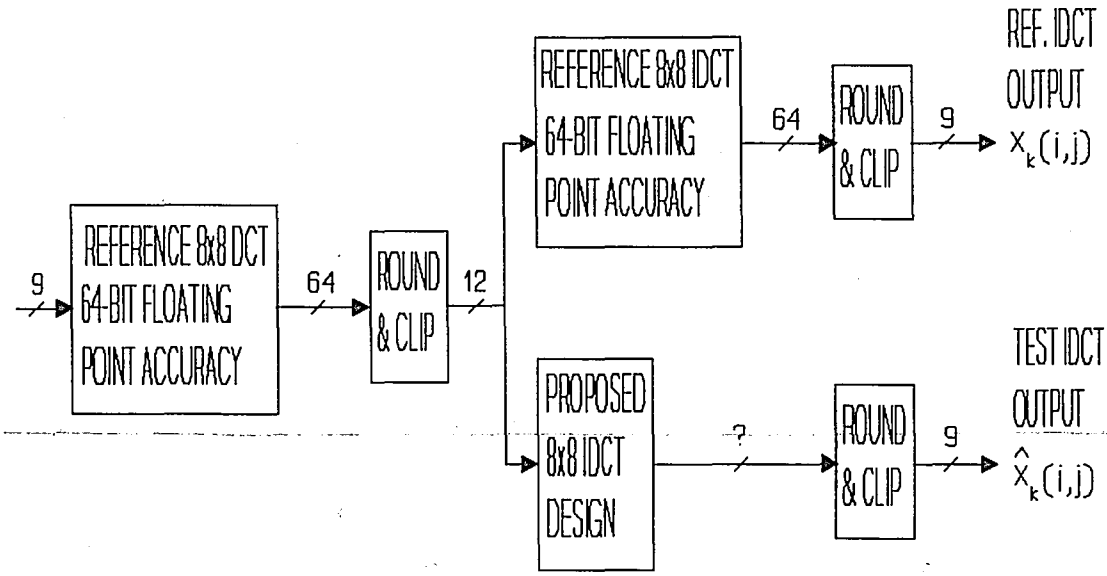

Figure 2. Testing procedure block diagram for a proposed 8x8 IDCT implementation.

explained in [11]. See Fig. 2 for an illustration [11].

1) Generate random integer pixel data values in the range -L to +H according to the random number generator (in C-Language) [11]. Arrange into 8x8 blocks by allocating each set of consecutive 8 numbers in a row. Data sets of 10,000 blocks each should be generated for (L=256,H=255), (L=H=5), (L=H=300).

2) For each 8x8 block, perform a separable, orthogonal, matrix multiply DCT, defined in (31), using at least 64-bit floating point accuracy.

3) For each 8x8 block, round the 64 resulting transformed coefficients to the nearest integer values. Then clip them to the range -2048 to +2047. This is the 12-bit input data to the inverse transform.

4) For each 8x8 block of 12-bit data produced by step 3, perform a separable, orthogonal, matrix multiply IDCT, defined in (32), using at least 64-bit floating point accuracy. Round the resulting pixels to the nearest integer, and clip them to the range -256 to +255. These blocks of 8x8 pixels are the "reference" IDCT output data.

5) For each 8x8 block of 12-bit data produced by step 3, use the proposed IDCT chip, or an exact-bit simulation thereof, to perform an IDCT. Clip the output to the range -256 to +255. These blocks of 8x8 pixels are the "test" IDCT output data.

6) For each of the 64 IDCT output pixels and for each of the 10,000 block data sets generated above, measure the peak, mean, and mean square errors between the "reference" data and the "test" data.

7) Rerun the measurements using exactly the same data values of step 1, but change the sign on each pixel. (Note: The resulted test data sets are in the ranges (-255,256), (-5,5), and (-300,300), respectively.)

DCT:

$$X(u,v) = \left(\frac{1}{4}\right) C(u)\ C(v) \sum_{i=0}^{7} \sum_{j=0}^{7} x(i,j) \cdot \cos\left(\frac{(2i+1)u\pi}{16}\right) \cos\left(\frac{(2j+1)v\pi}{16}\right) \tag{32}$$

IDCT:

$$x(i,j) = \left(\frac{1}{4}\right) \sum_{u=0}^{7} \sum_{v=0}^{7} C(u)\ C(v)\ X(u,v) \cdot \cos\left(\frac{(2i+1)u\pi}{16}\right) \cos\left(\frac{(2j+1)v\pi}{16}\right) \tag{33}$$

where x(i,j) (for i, j = 0, 1, ..., 7) is the pixel value,

X(u,v) (for u, v = 0, 1, ..., 7) is the transformed coefficient, C(0) = $1/\sqrt{2}$, and C(u) = C(v) = 1 (for u, v = 1, 2, ...,7).

Error measurements in step 6:

1) Error ($e_k(i,j)$) is defined as:

$$e_k(i,j) = \hat{x}(i,j) - x_k(i,j) \tag{34}$$

where k = 1, 2, ..., 10000.

2) Peak error (ppe(i,j)), at pixel location (i,j), is the peak value of $e_k(i,j)$ for k = 1, 2, ..., 10000.

3) Pixel mean square error (pmse(i,j)), at pixel location (i,j):

$$pmse(i,j) = \frac{\sum_{k=1}^{10000} e_k^2(i,j)}{10000}. \tag{35}$$

4) Overall mean square error (omse):

$$omse = \frac{\sum_{i=0}^{7}\sum_{j=0}^{7}\sum_{k=1}^{10000} e_k^2(i,j)}{64 \times 10000}. \tag{36}$$

5) Pixel mean error (pme(i,j)), at pixel location (i,j):

$$pme(i,j) = \frac{\sum_{k=1}^{10000} e_k(i,j)}{10000}. \tag{37}$$

6) Overall mean error (ome):

$$ome = \frac{\sum_{i=0}^{7} \sum_{j=0}^{7} \sum_{k=1}^{10000} e_k(i,j)}{64 \times 10000}. \qquad (38)$$

Specifications on the 8x8 IDCT accuracy measurements:

The following is the standard specifications that the 8x8 IDCT implementation must meet.

$$|ppe(i,j)| \leq 1$$

$$pmse(i,j) \leq 0.06$$

$$omse \leq 0.02$$

$$|pme(i,j)| \leq 0.015$$

$$|ome| \leq 0.0015$$

Also, for an input sequence of all zeros, the 8x8 IDCT output shall be all zeros.

The errors defined above are separated into two types; a "peak" error type (ppe(i,j), pmse(i,j), and pme(i,j)) and an "overall" error type (omse and ome). Peak error defines an error associated with particular pixel location (i,j). The testing procedure uses 10,000 8x8 pixel blocks. Thus, there are 10,000 tested pixels for each pixel location (i,j). Overall error defines the error of the entire system in which there are 640,000 tested pixels (8x8x10000).

The peak pixel error specification ($|ppe(i,j)| \leq 1$) implies, for any pixel location (i,j), that the difference between the proposed IDCT test model and the floating point

reference can be a maximum of 1. If, for example, the floating point reference pixel location ppe(0,0) is equal to 100, then the proposed IDCT test model result must be either 99 or 101.

The peak mean square error specification ($|pmse(i,j)| \leq 0.06$) implies that a maximum of 600 errors can occur, at any pixel location (i,j), over a 10,000 block test.

The overall mean square error specification ($omse \leq 0.02$) implies that a maximum of 12,800 errors, overall, can occur per 10,000 block test.

The peak mean error specification ($|pme(i,j)| \leq 0.015$) implies, for any pixel location (i,j), that if more than 150 errors occur (but less than 600 errors), then the errors must be positive and negative so as to cancel each other. For example, if the maximum number of errors (600) occurs at pixel location (0,0), then, in order for the pme(0,0) specification to be satisfied, a total of 375 positive errors and 225 negative errors must occur (375-225=150), or vise-versa.

The overall mean error specification ($|ome| \leq 0.0015$) implies that the overall errors, over the 10,000 block test, must be positive and negative so that the absolute value of their sum is 960 maximum.

The above procedure, "IEEE Standard Specifications for the Implementations of 8x8 IDCT", should be applied by the designers of IDCT chips. These are stringent requirements

that are very demanding of the designer. It is unlikely that the specifications can be satisfied without any regard to this procedure.

CHAPTER 5: HARDWARE IMPLEMENTATIONS

As mentioned, a two-dimensional DCT/IDCT micro chip (the "original design") has recently been deigned by another researcher [6]. This design implements the new algorithm, derived by Dr. Weiping Li (Chapter 3), with minimal hardware. However, the accuracy testing procedure, outlined in Chapter 4, was not applied during the design phase. The initial objective of this thesis was to build an exact-bit computer model of the original design, and to apply the "IEEE Standard Specifications for the Implementations of 8x8 Inverse Discrete Cosine Transform." Having carried out these tasks, it is rendered that the "original design" fails in satisfying the standard specifications. Even slight modifications, which increase the accuracy of the "original design", also fail.

As a result of these findings, two distinct two-dimensional DCT/IDCT implementations, which significantly satisfy the IEEE standard specifications, are introduced. Both designs presented here, implement an identical design structure as the "original design". However, the standard specifications are satisfied by increasing the sizes of the components and by "conditioning" the output.

The DCT/IDCT, as stated, is a separable transform, which means that a two-dimensional (2-D) DCT/IDCT can be implemented with two one-dimensional (1-D) DCT/IDCT processors in series. The new algorithm shows that a 1-D

DCT/IDCT consists of an 8x4 unit, a 4x2 unit, a 2x1 unit, a scaler unit, a preadd unit (for DCT only), and a postadd unit (for IDCT only). Figures 3 & 4 show 1-D DCT/IDCT Processing Units #1 and #2, respectively. The Preadd Unit,

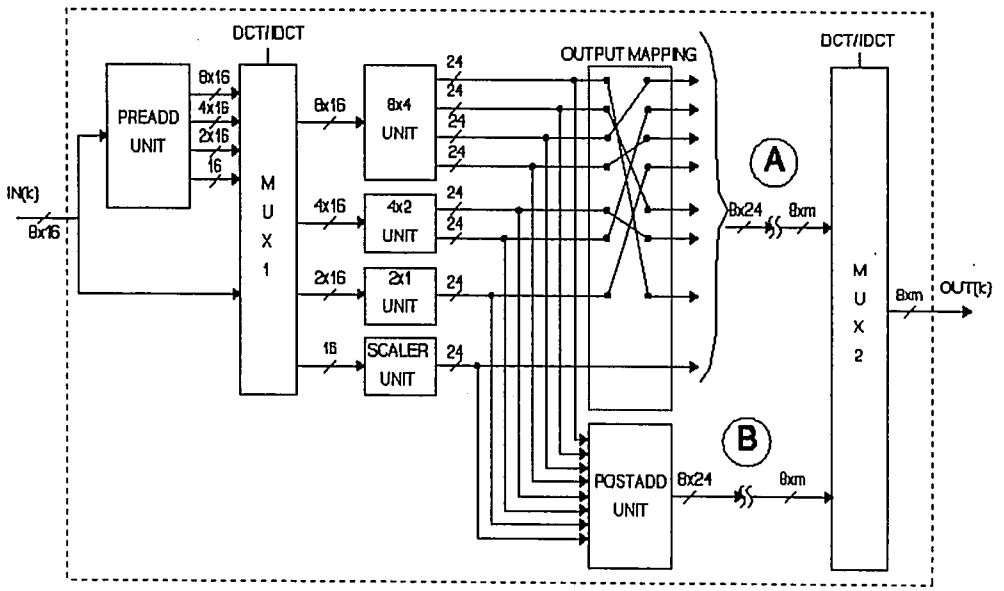

Figure 3. Unit #1: 1-D DCT/IDCT Processing Unit.

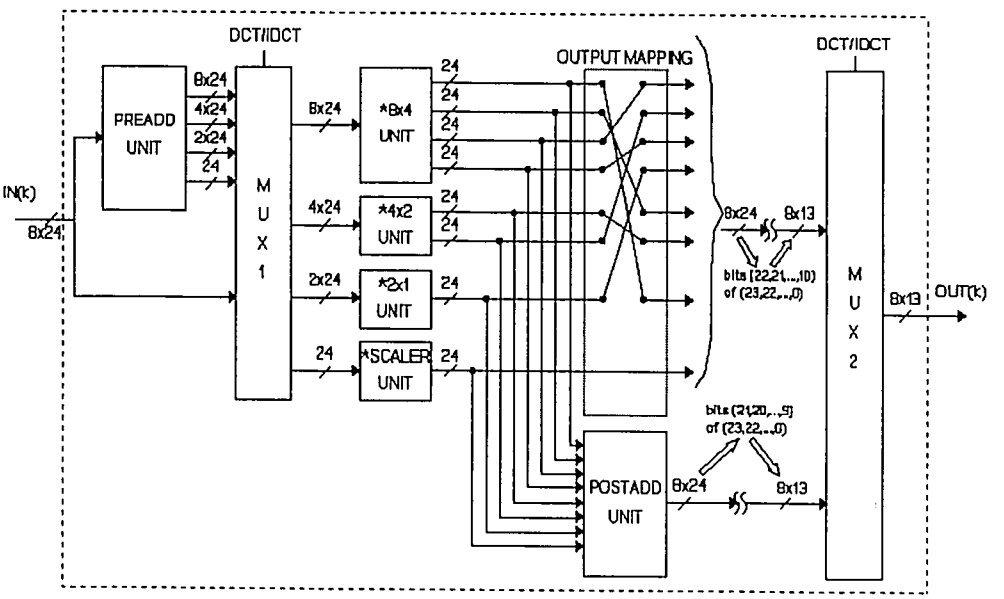

Figure 4. Unit #2: 1-D DCT/IDCT Processing Unit.

Postadd Unit, and multiplexers are identical for Processing Units #1 and #2. However, the 8x4, 4x2, 2x1, and Scaler Units, which perform the skew-circular convolution defined in the new algorithm, are different. Unit #1 inputs eight 16-bit words and performs 16 partial product generations and additions, in eight stages, to generate output. Whereas, Unit #2 inputs eight 24-bit words and performs 24 partial product generations and additions, in eight stages, to generate output. This is carried out by special skew-circular convolution units *8x4, *4x2, *2x1, *scaler. Refer to Appendix A for detailed schematic diagrams of the components featured in this design.

The outputs of both units are actually scaled due to the dynamic range of the DCT (≤ 4) and IDCT (≤ 2.78). This prevents overflow from occurring during computation. As a result, the units are designed such that the output for the DCT is scaled by a factor of eight (3-bits), and the IDCT is scaled by a factor of four (2-bits).

Figures #4 and #5 show Design #1 and Design #2 2-D DCT/IDCT Processors, respectively. An overview of the operation of both designs is as follows. The DCT and IDCT computations are performed on an 8x8 block of data (12-bit words). The 8x8 block is input one row at a time. Both the input and output are 12-bit words. However, for the DCT computation, 9-bit input words (range: -256 to +255) are used to generate 12-bit output words (range: -2048 to

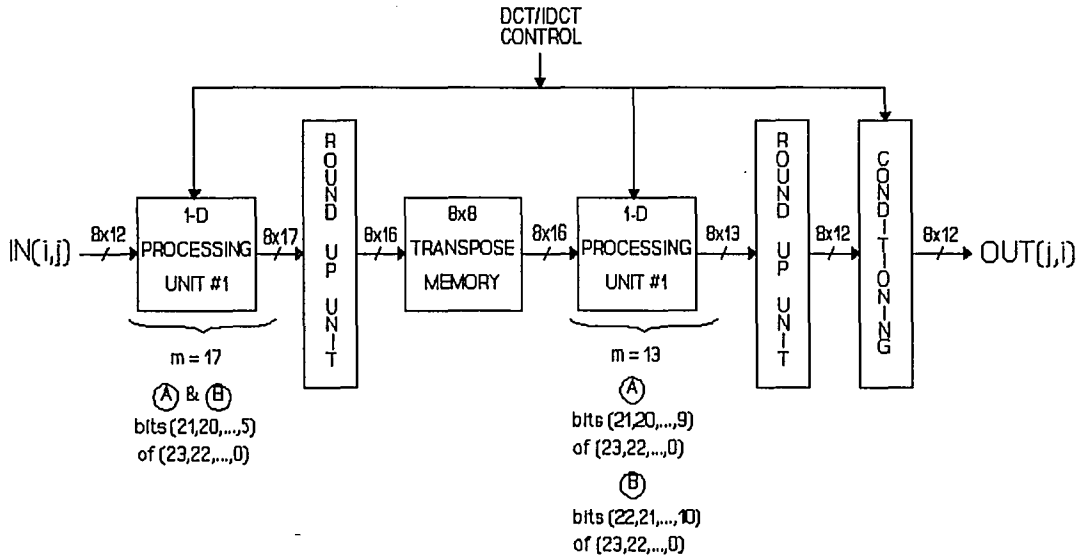

Figure 5. Design #1: DCT/IDCT Processor.

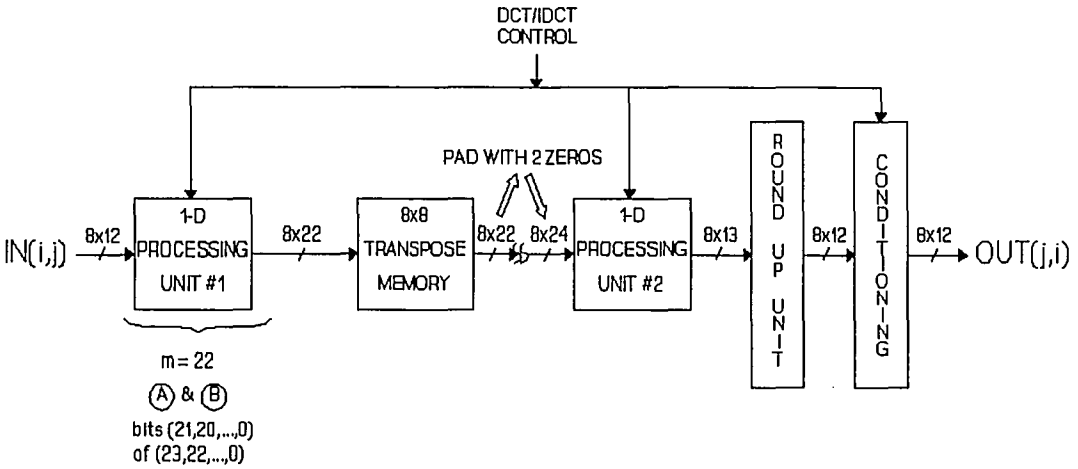

Figure 6. Design #2: 2-D DCT/IDCT Processor.

+2047). And, for the IDCT computation, 12-bit input words generate 9-bit output words. The 9-bit output words are the lower 9-bits of the 12-bit output that the dual DCT/IDCT processor generates.

Design #1

This design features two Unit #1 1-D DCT/IDCT processors in series. Remember that Unit #1 performs 16 partial product generations, and additions, to generate output. For the first 1-D processor (stage one), m = 17, which corresponds to the internal component sizing and the output bit size. In the 1-D unit, the output is clipped from 24-bits to m = 17-bits. These 17-bit output words, from stage one, are rounded up base on the least significant bit to generate a 16-bit word. The 8x8 block is collected and transposed. The transposed rows are input into the second Unit #1 1-D DCT/IDCT processor. As shown, m = 13 and the same output round-up scheme, explained for stage one, applies here. The output, following the second stage round-up unit, is then conditioned. This conditioning unit, since it pertains to both Designs #1 and #2, is examined latter.

Design #2

This design features 1-D DCT/IDCT processor Unit #1 (stage one) in series with Unit #2 (stage two). Because the input word length to the 2-D system is 12-bits, Unit #1, which handles 16 partial product additions, is used in stage one. However, what makes this design different from Design #1 is the fact that 22-bits (m = 22-bit for Unit #1) are transposed and input to stage two. Because of scaling, the first two bits, from the 24-bit output of Unit #1, are discarded (see Figure 3). Remember, Unit #2 1-D DCT/IDCT

processor performs 24 partial product generations and additions to generate output. This enhances the accuracy of Design #2, because no bits are discarded between the two stages, as in Design #1. Also, Design #2 only has a round-up unit following stage two since the 22-bit input to stage two, which enhances accuracy, eliminates the need to round-up.

Conditioning The Output

For both 2-D DCT/IDCT Processor Designs #1 and #2, the output of the second stage, is conditioned. Figure 7 shows the Conditioning Unit. When the DCT computation is performed, the output is unconditioned. However, when the IDCT computation is performed, this unit performs several functions.

One function is to prevent roll-over from occurring

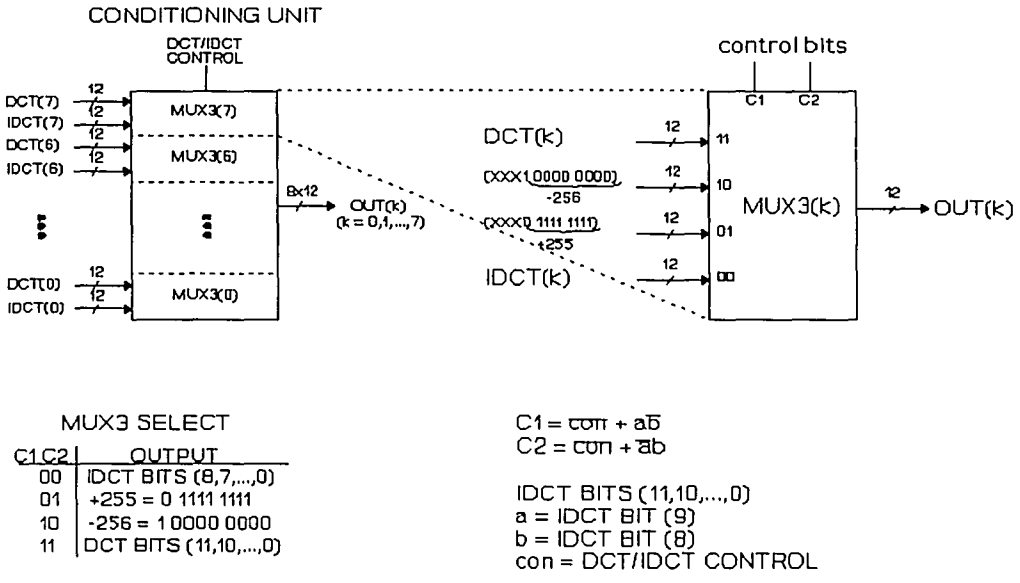

Figure 7. Conditioning Unit for 2-D DCT/IDCT processor.

which is caused by ROM content precision error and bit truncation in the processors, the problem of roll-over exists. An example is when the correct 9-bit IDCT output is 1 0000 0000 (-256), but the actual is 0 1111 1111 (+255). The absolute error, or difference between the two, is 512, when in fact, the correct result is achieved by simply adding 0 1111 1111 + 1 = 1 0000 0000. The way in which this is handled is explained shortly.

The other function this unit performs is to force the output to its maximum range, either -256 or +255 (9-bits), when the input data generates results outside this range. The IEEE testing procedure, outlined in Chapter 4, requires a random number test set of ±300 to be used, as input, to generate the DCT coefficients. The DCT results are clipped to the range -2048 to +2047 (12-bits) and used, as input, to generate the IDCT coefficients. Here, the results are clipped to the range -256 to +255 (9-bits). This is all done in a floating point model. Just the same, the actual processor needs to clip its results too.

The roll-over and the clipping are performed in the same step. The following shows how the system detects roll-over and when an output needs to be clipped. Remember that the IDCT output is 9-bits which corresponds to the range -256 to + 255. Here are some examples of results that do not require conditioning and some that do. The decimal and bit representations are shown. A tenth bit (in parentheses)

is also shown and serves as the detection bit.

Non-conditioned results:

+255 = (0) 0 1111 1111 -256 = (1) 1 0000 0000

+212 = (0) 0 1101 0100 -212 = (1) 1 0010 1100

Conditioned results:

+255 = (1) 0 1111 1111 is forced to -256 = 1 0000 0000

-300 = (1) 0 1101 0100 is forced to -256 = 1 0000 0000

-256 = (0) 1 0000 0000 is forced to +255 = 0 1111 1111

+300 = (0) 1 0010 1100 is forced to +255 = 0 1111 1111

The conditioning unit checks to see if the tenth bit and the ninth bit (MSB) are the same. If they are the same, it means that clipping is not required and that roll-over has not occurred. However, if these bits are different, then it means that the result requires conditioning based on the logic in Figure 7. The reason why the detection works on these two bits is because of scaling. The 9-bit result alone gives the unscaled result, whereas, the 10-bit result is scaled by two. Thus, these bits will be the same if roll-over does not occur, and the 9-bit range is not exceeded. In addition, when the 9-bit range (-256 to +255) is exceeded, these bits will be different.

This concludes the introduction of Design #1 and #2 2-D DCT/IDCT Processors. Refer to Appendix A for schematic diagrams of the designs.

CHAPTER 6: ACCURACY STUDY OF THE DCT/IDCT PROCESSOR DESIGN #1 AND DESIGN #2

The IEEE accuracy testing procedure and standard specifications, outlined in Chapter 4, has been applied to the 2-D DCT/IDCT processors (Design #1 and Design #2) presented in the previous chapter. The procedure requires that six sets of 10,000 8x8 random integer pixel values be generated according to the random number generator. The random number generator function, random(L,H), creates data sets of 10,000 blocks for the range -L to +H. The six data set ranges are as follows: positive random(256,255), positive random(5,5), and positive random(300,300) according to Step 1 of the procedure: and negative random(256,255), negative random(5,5), and negative random(300,300) according to Step 7 of the procedure. This translates to set ranges of (-256,255), (-5,5), (-300,300), (-255,256), (-5,5), and (-300,300).

Having established the notation to distinguish between the six sets, the results of the accuracy testing procedure are presented and compared against the standard specifications.

Peak mean square error (pmse(i,j)) results

The results of the six peak mean square error (pmse(i,j)) tests are shown in Tables 1 through 6 for both Designs #1 and #2. The entries in the tables are scaled up by 10^4 and should be multiplied by 10^{-4} to give the unscaled result. The specification requires that pmse(i,j) $\leq$ 0.06

(= $600 \cdot 10^{-4}$). Thus, keeping units in mind, each pmse(i,j) result shown must less than or equal to 600 in order for the specification to be satisfied.

In each of the tables, the largest entry of the 8x8 pmse(i,j) block is highlighted with a box. This serves as a marker to quickly compare the two designs against the standard specification (≤ 600). The tables show that none of the highlighted entries even comes close to approaching 600. Out of the six 8x8 pmse(i,j) tests for Design #1, the largest pmse(i,j) result is 120 (Table 2). And, for Design #2, the largest pmse(i,j) result is 46 (Table 4). Thus, both designs far exceed the peak mean square error specification. In comparing Design #1 against Design #2, it is shown, by adjacent 8x8 pmse(i,j) blocks, that Design #2 produces fewer errors overall. This will be more apparent in the overall mean square error results.

DESIGN #1

76	88	81	98	89	112	95	106
103	89	80	99	102	100	81	93
96	84	89	82	90	102	98	92
97	78	108	97	84	101	88	101
79	90	92	96	82	89	99	115
79	98	93	74	79	94	104	72
87	94	81	103	87	86	85	84
116	88	87	89	107	87	89	98

DESIGN #2

32	35	14	8	15	2	3	2
4	6	4	12	3	15	8	11
12	2	11	3	9	1	6	3
5	14	11	8	11	10	15	9
7	6	6	5	2	3	4	9
14	11	10	5	10	11	7	11
6	4	3	5	6	3	8	6
4	3	5	5	4	3	4	7

Table 1. Peak mean square error results (pmse(i,j)) for the positive random(256,255) range. Note: unit is 10^{-4}.

DESIGN #1							
79	82	89	89	82	118	102	105
106	92	88	103	91	95	84	102
92	90	99	85	85	99	98	90
104	79	106	96	80	99	91	104
75	97	95	92	80	88	97	**120**
82	103	96	82	82	95	107	82
95	95	79	105	81	84	85	74
105	84	90	97	109	101	92	90

DESIGN #2							
23	**31**	25	1	19	2	4	11
1	4	6	12	7	15	10	11
7	6	9	4	6	4	0	1
14	9	16	10	9	14	9	16
7	6	9	6	4	5	2	7
4	7	12	10	5	10	16	12
1	4	6	2	5	7	2	2
9	5	5	7	6	2	6	2

Table 2. Peak mean square error results (pmse(i,j)) for the negative random(256,255) range. Note: unit is 10^{-4}.

DESIGN #1							
97	91	**112**	96	91	73	92	72
103	111	94	99	86	81	101	102
107	96	93	92	92	99	91	82
76	77	98	87	91	89	80	99
93	78	98	95	99	95	111	108
97	90	93	70	83	85	90	88
94	75	98	87	82	88	101	87
84	83	89	88	93	107	80	88

DESIGN #2							
42	33	32	9	19	0	9	8
3	4	5	7	3	4	9	9
6	11	10	5	7	4	9	5
12	5	7	6	13	12	10	8
4	13	5	3	4	6	4	5
11	9	15	12	5	14	13	3
5	1	2	5	4	13	8	4
5	4	6	6	8	2	4	3

Table 3. Peak mean square error results (pmse(i,j)) for the positive random(5,5) range. Note: unit is 10^{-4}.

DESIGN #1							
90	97	**108**	96	96	82	90	76
104	107	102	92	90	83	93	97
100	96	95	88	91	102	84	88
65	70	95	91	89	92	68	101
92	87	94	91	97	84	106	106
97	84	90	73	82	84	92	97
95	72	101	80	78	79	91	96
97	77	90	78	94	109	78	88

DESIGN #2							
35	**46**	18	4	19	4	7	7
2	1	1	8	4	13	9	11
9	14	9	3	10	7	5	3
9	13	11	6	4	6	12	14
6	8	4	1	5	3	5	4
17	9	11	6	7	11	14	12
5	1	6	4	7	6	2	5
3	2	7	7	5	4	7	6

Table 4. Peak mean square error results (pmse(i,j)) for the negative random(5,5) range. Note: unit is 10^{-4}.

DESIGN #1							
86	84	85	79	82	71	83	86
67	81	77	77	80	87	83	76
66	95	94	78	80	72	77	82
75	64	77	61	85	78	64	74
74	85	71	71	65	64	79	79
79	83	73	84	69	67	63	71
78	76	75	75	89	92	66	79
82	78	77	71	81	71	63	64

DESIGN #2							
35	34	29	4	27	8	6	7
3	2	9	10	4	13	6	9
11	11	7	5	7	1	1	6
13	8	12	6	15	4	8	7
5	1	2	1	2	3	2	2
15	9	7	12	10	2	4	6
5	3	3	5	3	5	6	1
4	2	7	4	4	6	4	5

Table 5. Peak mean square error results (pmse(i,j)) for the positive random(300,300) range. Note: unit is 10^{-4}.

DESIGN #1							
81	86	84	81	85	74	90	86
77	84	77	69	79	82	82	81
72	86	96	75	72	70	70	86
69	72	85	68	83	71	76	75
77	91	75	72	67	68	75	78
66	85	87	85	78	79	61	87
82	71	75	81	91	89	65	75
78	82	74	74	82	69	68	64

DESIGN #2							
34	32	18	8	14	5	4	3
5	3	6	5	8	4	12	9
11	4	7	1	7	3	2	6
10	13	7	7	8	10	9	8
6	5	8	0	2	3	2	1
3	3	5	8	7	9	8	6
5	5	3	8	2	6	6	4
3	6	1	2	5	5	2	1

Table 6. Peak mean square error results (pmse(i,j)) for the negative random(300,300) range. Note: unit is 10^{-4}.

Overall mean square error (omse) results

Table 7 shows the overall mean square error (omse) results for 2-D DCT/IDCT Processor Designs #1 and #2. Both designs exceed the standard specification, omse ≤ 0.02. As expected, from observation of the peak mean square error results, Design #1 has a larger omse result than Design #2; more than a magnitude greater (x10).

Data Set Range	omse Design #1	omse Design #2
(+) (256,255)	0.009191	0.000783
(-) (256,255)	0.009286	0.000801
(+) (5,5)	0.009136	0.000831
(-) (5,5)	0.009027	0.000819
(+) (300,300)	0.007656	0.000747
(-) (300,300)	0.007773	0.000661

Table 7. Overall mean square error (omse) results, where omse ≤ 0.02 is the standard specification.

Peak mean error (pme(i,j)) results

Tables 8 through 13 show the results of the peak mean error (pme(i,j)) tests for 2-D DCT/IDCT Processors Design #1 and #2. The entries in these tables are scaled by 10^4 (i.e the unit of the entries is 10^{-4}). The standard specification is $|pme(i,j)| \leq 0.015$ ($= 150 \cdot 10^{-4}$). Thus, the results in the tables must be less than or equal to 150. Again, the largest pme(i,j) of each 8x8 block is highlighted in an enclosed box for quick reference to the standard specification. For Design #1 and Design #2, the largest pme(i,j) is 34 (Table 8) and 46 (Table 11), respectively. Both designs, again, exceed the standard specification. The peak mean error basically indicates how well the design generates a balance of positive and negative 1 errors per pixel location.

DESIGN #1

26	[34]	3	26	3	26	3	0
9	19	6	25	4	2	13	17
14	14	15	12	16	0	22	12
7	16	4	21	0	3	8	15
21	18	10	8	4	3	5	17
9	8	9	4	5	12	6	2
7	2	1	9	15	4	15	4
10	6	21	11	5	11	3	10

DESIGN #2

32	[35]	14	8	15	2	3	2
2	4	2	12	3	15	8	9
12	2	11	1	9	1	6	1
5	14	11	8	11	10	15	9
7	6	6	3	2	3	2	1
14	11	10	5	10	11	7	11
2	2	3	5	2	3	6	6
4	3	5	5	2	3	2	7

Table 8. Peak mean error results (pme(i,j)) for the positive random(256,255) range. Note: unit is 10^{-4}.

DESIGN #1

19	14	[31]	17	18	10	14	7
4	6	10	1	19	7	16	12
4	6	5	7	9	11	14	14
28	1	30	2	16	11	7	0
11	25	17	10	0	0	5	18
14	9	4	12	8	15	15	14
11	5	11	3	21	6	11	4
3	2	14	3	3	13	14	4

DESIGN #2

23	[31]	25	1	19	2	4	11
1	1	6	12	0	15	10	11
7	6	4	1	4	1	0	1
14	9	16	10	9	14	9	16
7	4	9	1	4	2	0	2
4	7	12	10	5	10	16	12
1	1	6	2	2	7	2	2
1	2	2	7	6	2	4	2

Table 9. Peak mean error results (pme(i,j)) for the negative random(256,255) range. Note: unit is 10^{-4}.

DESIGN #1

23	23	14	18	7	23	18	4
1	23	2	3	12	7	23	6
21	12	5	0	6	13	5	18
10	7	2	5	15	13	10	1
3	4	[30]	11	5	5	13	12
1	18	13	22	9	3	16	6
14	21	4	5	8	14	11	7
6	7	1	4	13	3	18	10

DESIGN #2

[42]	33	32	9	19	0	9	8
1	2	3	7	1	4	9	9
6	11	6	5	7	2	7	3
12	5	7	6	11	12	8	8
4	13	5	1	2	4	4	3
11	9	15	12	5	14	13	3
5	1	0	3	2	13	8	2
5	2	4	4	6	2	4	1

Table 10. Peak mean error results (pme(i,j)) for the positive random(5,5) range. Note: unit is 10^{-4}.

DESIGN #1

24	29	14	20	12	10	12	6
2	19	14	20	4	17	1	13
12	16	9	8	19	10	0	6
9	12	27	11	7	2	8	15
8	11	18	13	3	8	10	10
15	6	0	7	2	6	2	21
1	10	15	12	12	7	17	14
1	9	2	8	2	17	**32**	14

DESIGN #2

35	**46**	18	4	19	4	5	7
2	1	1	8	4	13	9	11
7	14	9	1	10	5	1	3
9	13	11	6	4	6	12	14
6	8	4	1	3	3	3	2
17	9	11	6	7	9	14	12
3	1	4	4	3	4	0	5
3	2	5	5	5	2	7	4

Table 11. Peak mean error results (pme(i,j)) for the negative random(5,5) range. Note: unit is 10^{-4}.

DESIGN #1

14	14	5	**29**	10	15	5	4
11	13	1	17	10	25	5	22
2	7	2	2	8	4	3	2
5	12	17	3	15	6	2	8
4	7	17	1	3	0	5	15
23	19	21	14	5	3	1	7
14	6	1	1	19	4	10	5
8	4	3	3	1	13	3	0

DESIGN #2

35	34	29	4	27	8	4	7
1	2	9	10	2	13	6	9
11	11	7	3	7	1	1	4
13	8	12	6	15	4	8	7
5	1	2	1	2	1	0	2
15	9	7	12	8	2	4	6
5	1	1	5	1	5	4	1
4	2	5	2	2	6	4	5

Table 12. Peak mean error results (pme(i,j)) for the positive random(300,300) range. Note: unit is 10^{-4}.

DESIGN #1

27	**32**	20	1	3	8	4	4
1	12	3	3	19	8	16	9
10	14	12	3	16	0	2	6
15	12	5	4	3	1	14	1
3	11	23	2	5	4	9	8
6	1	3	5	14	13	5	19
6	9	5	19	21	17	7	9
4	14	4	8	4	7	4	4

DESIGN #2

34	32	18	6	14	5	4	3
3	3	2	5	8	4	12	9
11	4	7	1	7	1	2	0
10	13	7	7	8	10	9	8
6	5	6	0	2	3	0	1
3	3	5	8	7	9	8	6
5	3	3	6	2	4	6	4
3	6	1	2	5	5	2	1

Table 13. Peak mean error results (pme(i,j)) for the negative random(300,300) range. Note: unit is 10^{-4}.

Overall mean error (ome) results

Table 14 shows the overall mean error (ome) results for 2-D DCT/IDCT Processor Designs #1 and #2. Both designs exceed the standard specification, ome ≤ 0.0015.

Data Set Range	ome Design #1	ome Design #2
(+) (256,255)	0.000153	0.000002
(-) (256,255)	0.000123	0.000021
(+) (5,5)	0.000058	0.000041
(-) (5,5)	0.000220	0.000053
(+) (300,300)	0.000109	0.000053
(-) (300,300)	0.000086	0.000020

Table 14. Overall mean error (ome) results, where ome ≤ 0.0015 is the standard specification.

Peak pixel error (ppe(i,j)) and zero input response

Design #1 and Design #2 generate peak pixel error results (ppe(i,j)) that do not exceed the standard specification, |ppe(i,j)| ≤ 1. In addition, both designs generate zero output for zero input.

An error block of the "original design"

Table 15 shows one error block result (positive random(256,255) range) of the modified "original design". This is the difference between a single 8x8 block "test" IDCT result and "reference" IDCT result. This single block exceeds the peak pixel error (ppe(i,j)).

112	-23	62	64	-107	1	-117	223
31	-42	150	155	76	73	-72	13
21	-138	-105	-49	-114	3	-145	82
-66	-201	-79	-60	-31	12	-142	-133
-85	-56	-132	52	80	-42	4	-153
-70	38	-207	116	51	-100	31	-10
-128	58	-100	163	84	-178	29	-146
71	1	59	126	-46	-2	-94	183

Table 15. Error of one block ("test" IDCT minus "reference" IDCT) for the "original design".

CHAPTER 7: RESULTS & CONCLUSIONS

In summary, this thesis features two distinct 2-D DCT/IDCT processors that far exceed the "IEEE Standard Specifications for the implementations of 8x8 IDCT." The processors input and output 12-bit results. For the DCT computation, the 12-bit input is actually 9-bits with 3 zeros padded to the end, and the output is the 12-bit result. For the IDCT computation, however, the input is 12-bits and the output is the least significant 9-bits. Both designs incorporate the separable property of the DCT by implementing two 1-D DCT/IDCT processors in series to create the 2-D processor. The major difference between the two designs is the number of partial product generations and additions in the second 1-D DCT/IDCT processor; 16 for Design #1 and 24 for Design #2. In addition, for Design #1 only, the results generated by the first 1-D DCT/IDCT processor are rounded-up before entering the second stage. Both designs, however, use the round-up feature following the second 1-D stage. A special output conditioning scheme, which maintains continuity and accounts for roll-over, exists for both designs.

Both Designs successfully meet the standard specifications by incorporating 24-bit ROMs, 24-bit biased-redundant binary adders (BABA), 24-bit carry save adders (CSA), rounding of the outputs, and output conditioning. The "original design", however, fails tremendously in

meeting the specifications. In contrast, the "original design" incorporates 16-bit ROMs, 16-bit BRBAs and CSAs, no rounding of the output, and no output conditioning. Even modifying this design, with rounding and output conditioning, fails to decrease the errors in accuracy.

Designs #1 and Design #2 far exceed the standard specifications. Design #2 produces far less errors than Design #1. It is proposed, however, that Design #1 be used in practice over Design #2. This is because Design #1 is most similar to the "original design". As a consequence, Design #1 has the same number of pipelining stages as the "original design." If the design principles used for the layout of the "original design" are applied during the layout of Design #1, then a relatively identical operating frequency can be obtained. This is not the case for Design #2 since it has additional pipelining stages which will affect the propagation times. Design #2 is incorporated in this thesis to serve as a future implementation in case applications are developed that call for more stringent standard specifications.

REFERENCES

[1] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete Cosine Transform," IEEE Trans. Comput., (January 1974), pp. 90 - 93.

[2] K. R. Rao and P. Yip, Discrete Cosine Transform: Algorithms, Advantages, and Applications. San Diego: Academic Press, Inc., 1990.

[3] M. J. Narasimha and A. M. Peterson, "On the Computation of Discrete Cosine Transform," IEEE Trans. Commun., vol. COM-26, no. 6, pp. 934-936, June 1978.

[4] W. Chen et al., "A Fast Computational Algorithm for the Discrete Cosine Transform," IEEE Trans. Commun., vol. COM-25, no. 9, pp. 1004-1009, Sept. 1977.

[5] J. Makhoul, "A Fast Cosine Transform in One and Two Dimensions," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-28, no. 1, pp. 27-34, Feb. 1980.

[6] H. V. Sorensen et al., "Real-valued Fast Fourier Transform Algorithms," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-35, no. 6, pp. 849-863, June 1987.

[7] M. Vetterli and H. J. Nussbaumer, "Simple FFT and DCT Algorithms with Reduced Number of Operations," Signal Processing, vol. 6, no. 4, pp. 267-278, Aug. 1984.

[8] M. Buttner and H. W. Schussler, "On Structures for the Implementation of the Distributed Arithmetic," Nachrichtentech. Z., vol. 29, pp. 472-477, 1976.

[9] B. G. Lee, "A New Algorithm to Compute the Discrete Cosine Transform, " IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-32, no. 6, pp. 1243-1245, Dec. 1984.

[10] R. J. Clarke et al, "On the Dynamic Range of Cefficients Generated in Transform Processing of Digitised Image data," IEEE Proceedings, vol. 132, pt. F, no. 2, pp. 107-110, April 1985.

[11] IEEE Circuits and Systems Society, "IEEE Standard Specifications for the Implementations of 8x8 Inverse Discrete Cosine Transform," IEEE Std. 1180-1990, March 1991.

[12] W. Li, "A New Algorithm to Compute the DCT and Its Inverse," IEEE Trans. Signal Processing, vol. 39, no. 6, pp. 1305-1313, June 1990.

[13] D. Slawecki, "A High Speed 2-D DCT/IDCT Processor," M.S. thesis, Lehigh University, Aug. 1991.

APPENDIX A: ADDITIONAL HARDWARE COMPONENT DESIGN

Figure A1 shows the skew-circular convolution hardware configuration, called the DCT/IDCT Index "k" Processing Unit. Figure A1.a shows the symbol representation for the 16 partial product generations and additions hardware configuration shown in Figure A1.b (used in 1-D DCT/IDCT Processing Unit #1). Figure A1.c shows the symbol representation for the 24 partial product generations and additions hardware configuration (used in 1-D DCT/IDCT Processing Unit #2). For this case, the dotted box in Figure A1.b is replaced by the configuration in Figure A1.d. For the 8x4 Unit (and *8x4), k = {1,3,5,7} and m = 4. For the 4x2 Unit (and *4x2), k = {2,6} and m = 2.

Bit Shifting Scheme

In Figure A1.b and A1.d there are nodes that are labeled with an encircled letter. For each of these nodes, the bit shifting scheme is described. The bits of a component are denoted by a subscript, where, for example, out_{23} and out_0 represent the MSB and the LSB, respectively. Table A1 shows the bit shifting scheme for both configurations depicted in Figure A1. For the 24 partial product generations and additions configuration, nodes B, C, D, E, F, G, and H remain the same, however, node A is replaced with A1, A2, and A3. See Appendix B for ROM content description.

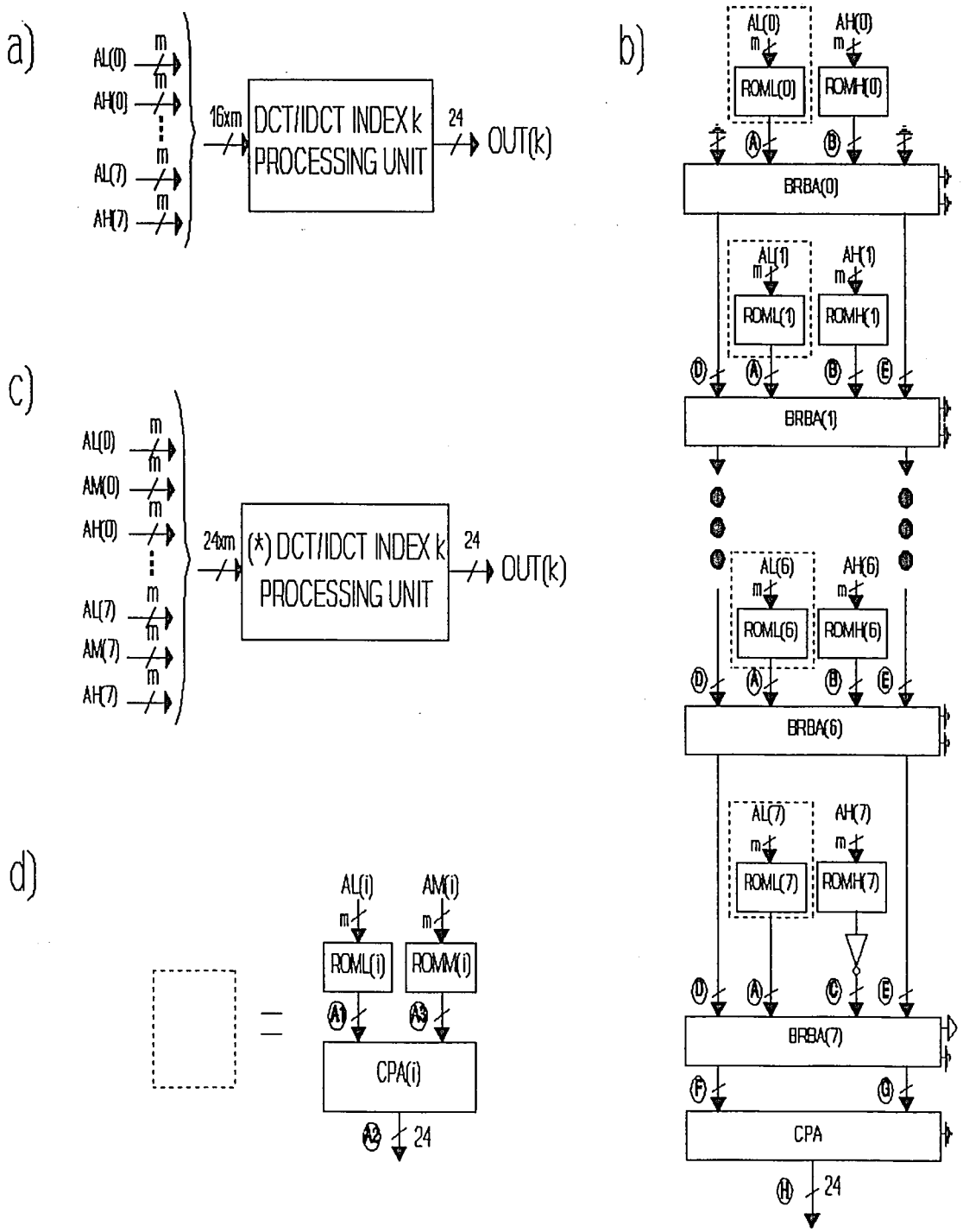

Figure A1. Skew-circular convolution DCT/IDCT "index k" Processing Unit.

NODE	OUTPUT TERMINAL	CONNECTED TO INPUT TERMINAL	i	j
A	ROML(i).out_{23} ROML(i).out_j	BRBA(i).$a2_{23}$ BRBA(i).$a2_{j-1}$	0,1,...,7	1,2,..., 23
B	ROMH(i).out_j	BRBA(i).$b1_j$	0,1,...,6	0,1,..., 23
C	ROMH(7).out_j	INVERT BRBA(7).$b1_j$		0,1,..., 23
D	BRBA(i-1).$outS1_{24}$ BRBA(i-1).$outS1_j$	BRBA(i).$a1_{23}$ BRBA(i).$a1_{j-2}$	1,2,...,7	2,3,..., 24
E	BRBA(i-1).$OUTS2_{24}$ BRBA(i-1).$outS2_j$	BRBA(i).$b2_{23}$ BRBA(i).$b2_{j-2}$	1,2,...,7	2,3,..., 24
F	BRBA(7).$outS1_j$	CPA.a_j		0,1,..., 23
G	BRBA(7).$outS2_j$	CPA.b_j		0,1,..., 23
H	CPA.out_j			0,1,..., 23
*	**********	**********	********	********
A1	ROML(i).out_{23} ROML(i).out_j	CPA(i).a_{23} CPA(i).a_{j-1}	0,1,...,7	1,2,..., 23
A2	ROMM(i).out_j	CPA(i).b_j	0,1,...,7	1,2,..., 23
A3	CPA(i).out_{23} CPA(i).out_j	BRBA(i).$a2_{23}$ BRBA(i).$a2_{j-1}$	0,1,...,7	1,2,..., 23

Table A1. Bit shifting scheme of the 8x4 and 4x2 units (nodes A through H). Bit shifting scheme of the *8x4 and *4x2 units with node A replaced by nodes A1, A2, and A3.

Figure A2 shows the hardware configuration for the 2x1 (*2x1) and the scaler (*scaler) DCT/IDCT Index "k" Processing Units, where k = {0,4}. For the 16 partial product scheme, m = 2, whereas, m = 3 for the 24 partial product scheme (denoted by *).

Bit Shifting Scheme

Table A2 shows the bit shifting scheme for the 16 and the 24 partial product schemes depicted in Figure A2. Unlike the bit shifting scheme in Table A1, one scheme applies to both the 16 and 24 partial product hardware configurations.

NODE	OUTPUT TERMINAL	CONNECTED TO INPUT TERMINAL	i	j
A	ROM(i).out$_j$	CSA(i).b$_j$	0,1,...,7	0,1,..., 23
B	CSA(i-1).outS$_{23}$ CSA(i-1).outS$_{23}$ CSA(i-1).outS$_j$	CSA(i).a$_{23}$ CSA(i).a$_{22}$ CSA(i).a$_{j-2}$	0,1,...,7	2,3,..., 23
C	CSA(i-1).outC$_{23}$ CSA(i-1).outC$_j$	CSA(i).c$_{23}$ CSA(i).c$_{j-1}$	0,1,...,7	1,2,..., 23
D	CSA(7).outS$_j$	CPA.a$_j$		0,1,..., 23
E	CSA(7).outC$_j$ CSA(6).outC$_0$	CPA.b$_{j+1}$ CPA.b$_0$		0,1,..., 22
F	CPA.out$_j$			0,1,..., 23

Table A2. Bit shifting scheme of the 2x1 and scaler units (and *2x1 and *scaler).

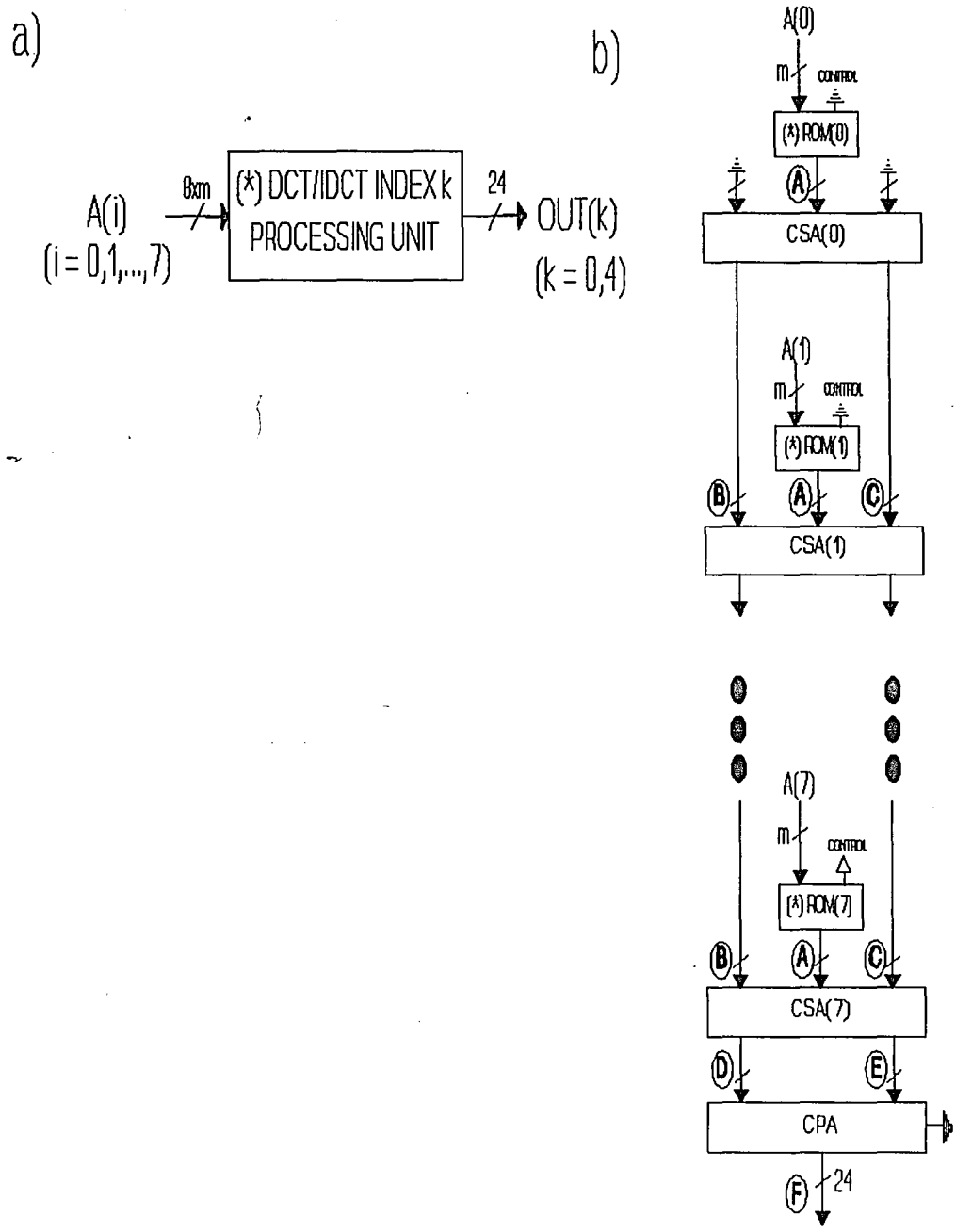

Figure A2. DCT/IDCT Index "k" Processing Unit used in the 2x1 (*2x1) and scaler (*scaler) units.

The remaining figures are detailed schematics of the components featured in the design. In all of the figures, the 24 partial product configuration, used in 1-D Processing Unit #2 (Figure 4), is denoted by an (*).

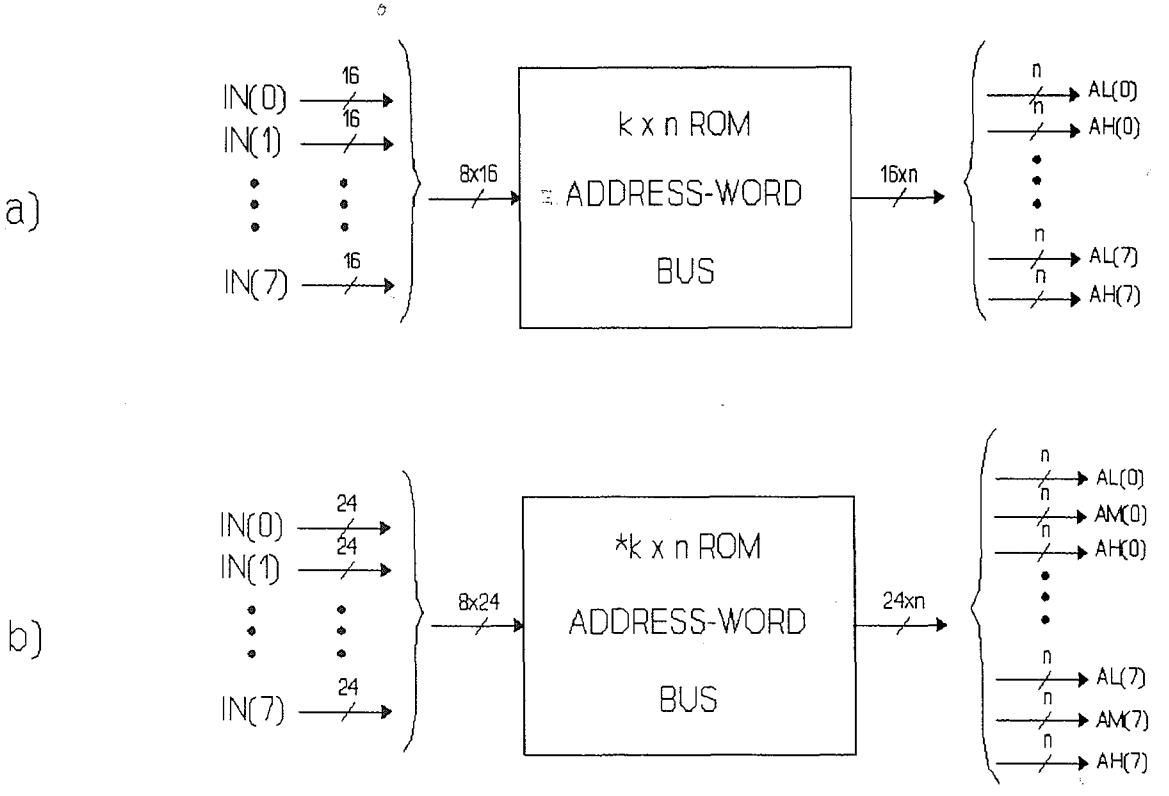

Figure A3. Symbol representation of the kxn (*kxn) ROM Address-Word Bus, where k = 8 & n = 4 for the 8x4 Unit, and k = 4 & n = 2 for the 4x2 Unit.

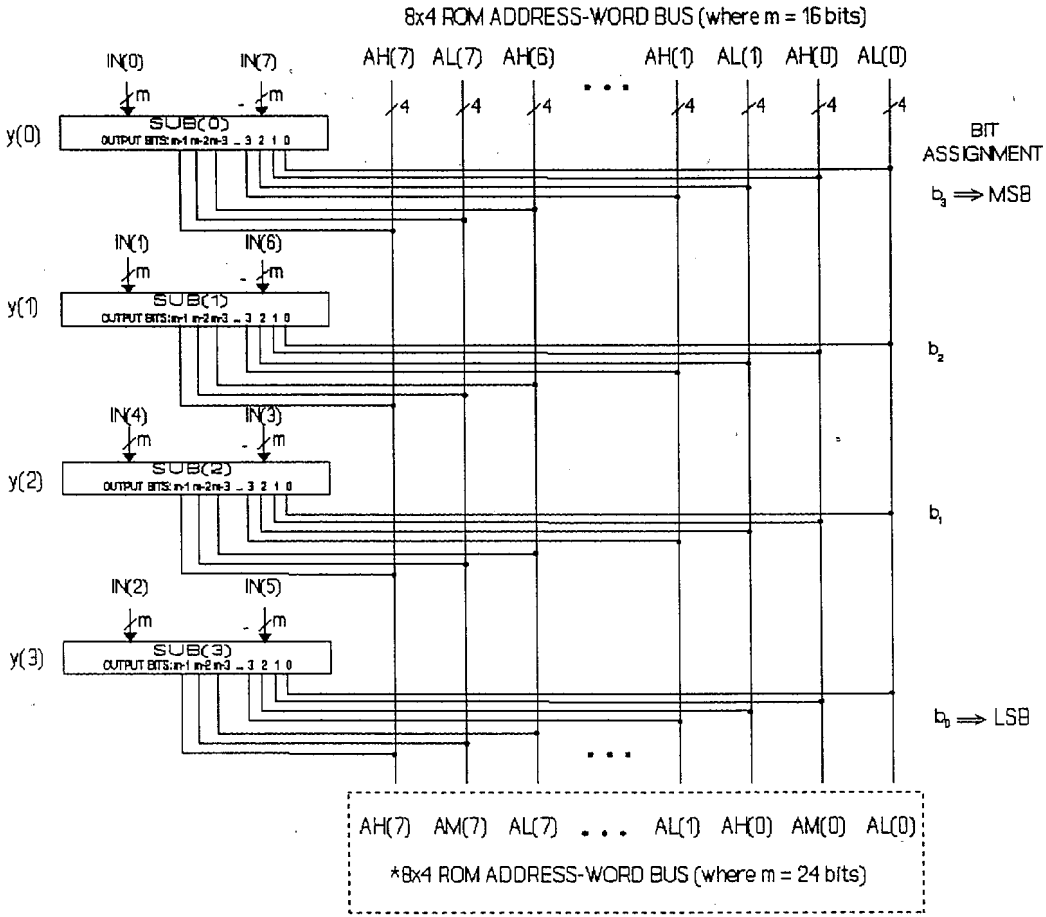

Figure A4. Hardware design of the 8x4 (*8x4) ROM Address-Word Bus.

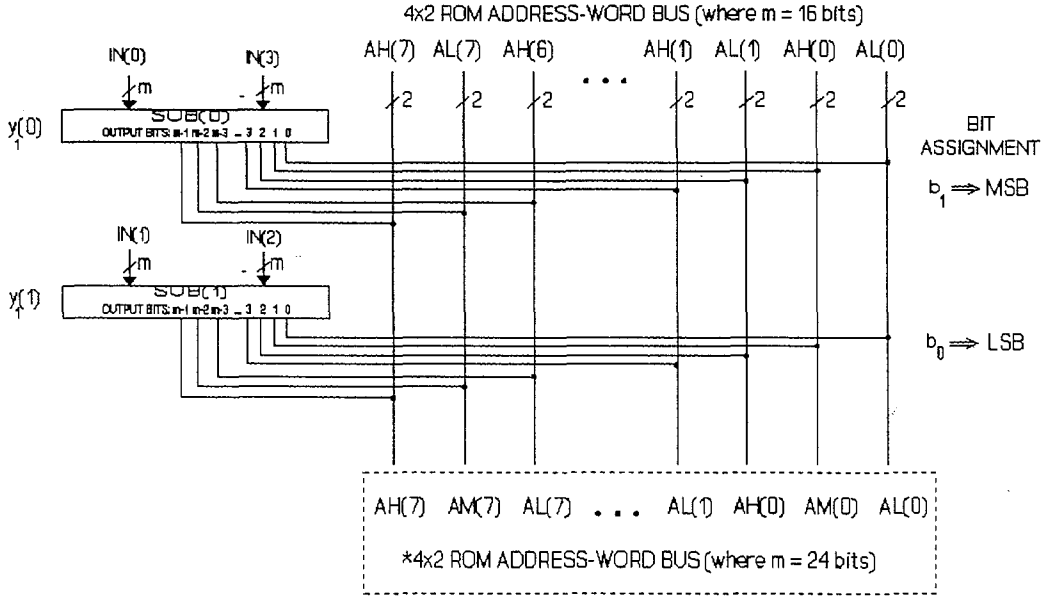

Figure A5. Hardware design of the 4x2 (*4x2) ROM Address-Bus.

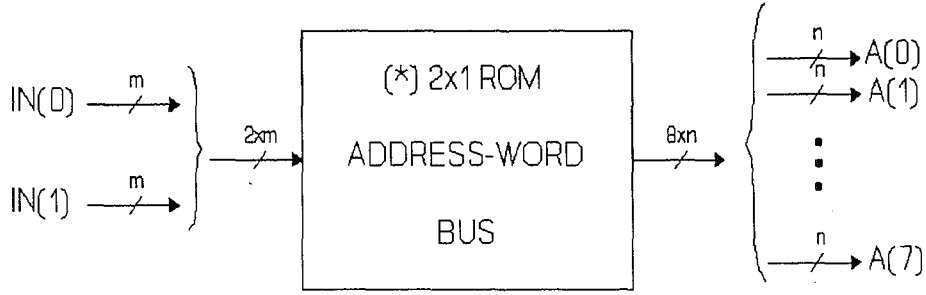

Figure A6. Symbol representation of the 2x1 (*2x1) ROM Address-Word Bus. Where m = 16 & n = 2 for the 2x1 Unit, and m = 24 & n = 3 for the *2x1 Unit.

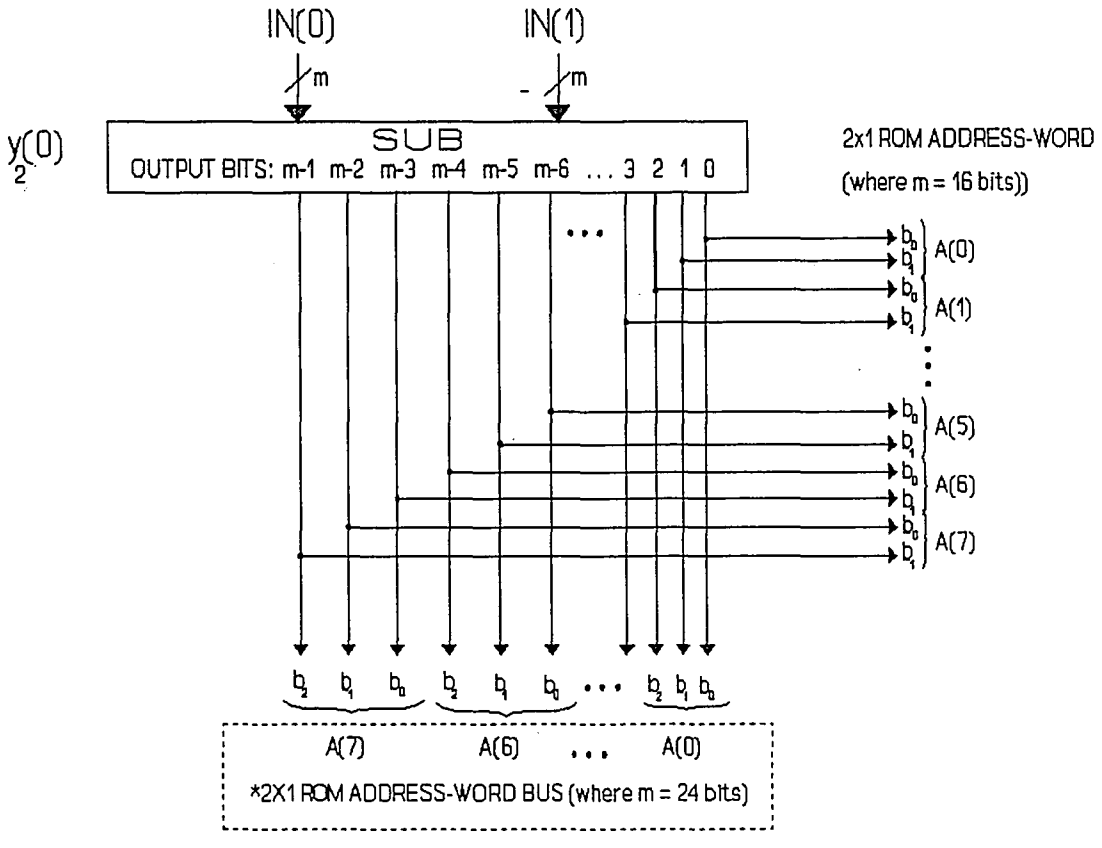

Figure A7. Hardware design of the 2x1 (*2x1) ROM Address-Word Bus.

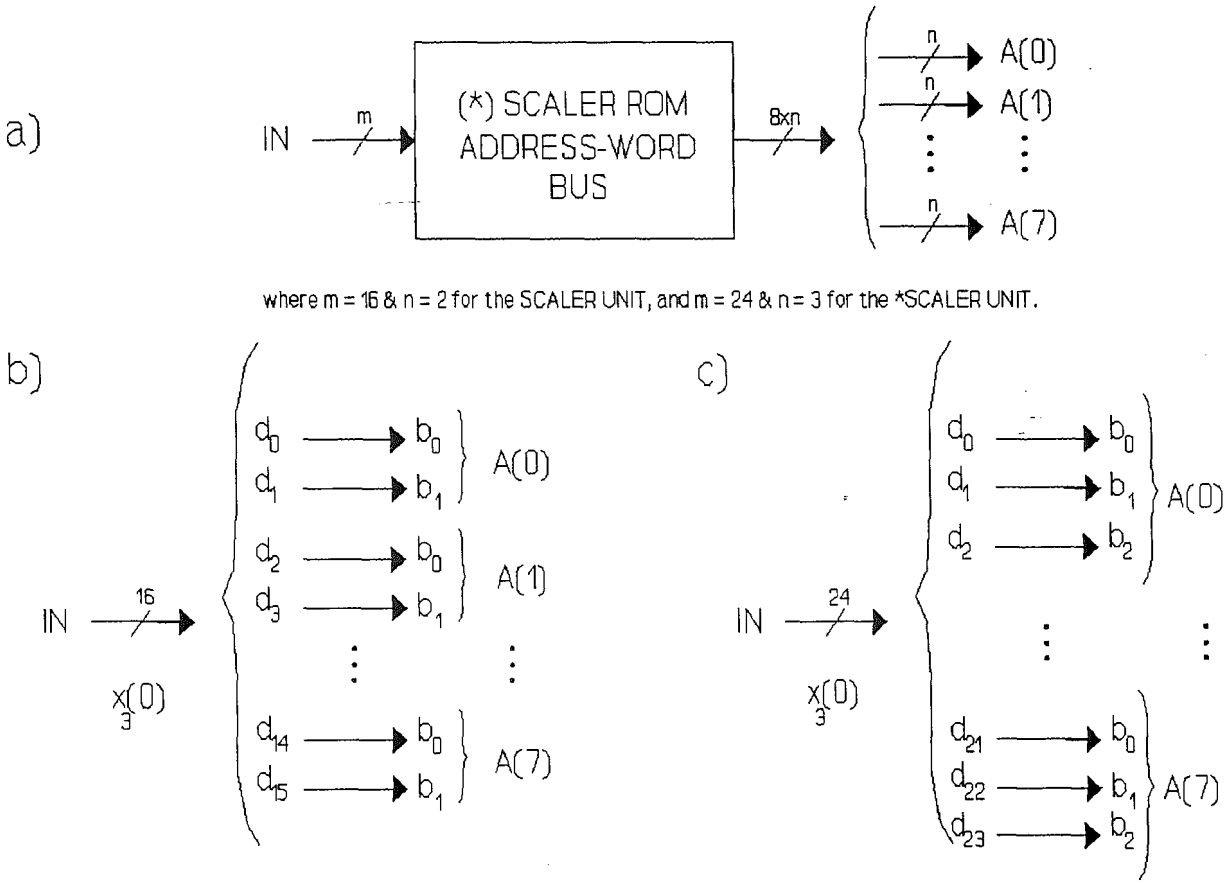

Figure A8. Symbol representation (a) and hardware connection scheme of the Scaler Unit (b) and *Scaler Unit (c).

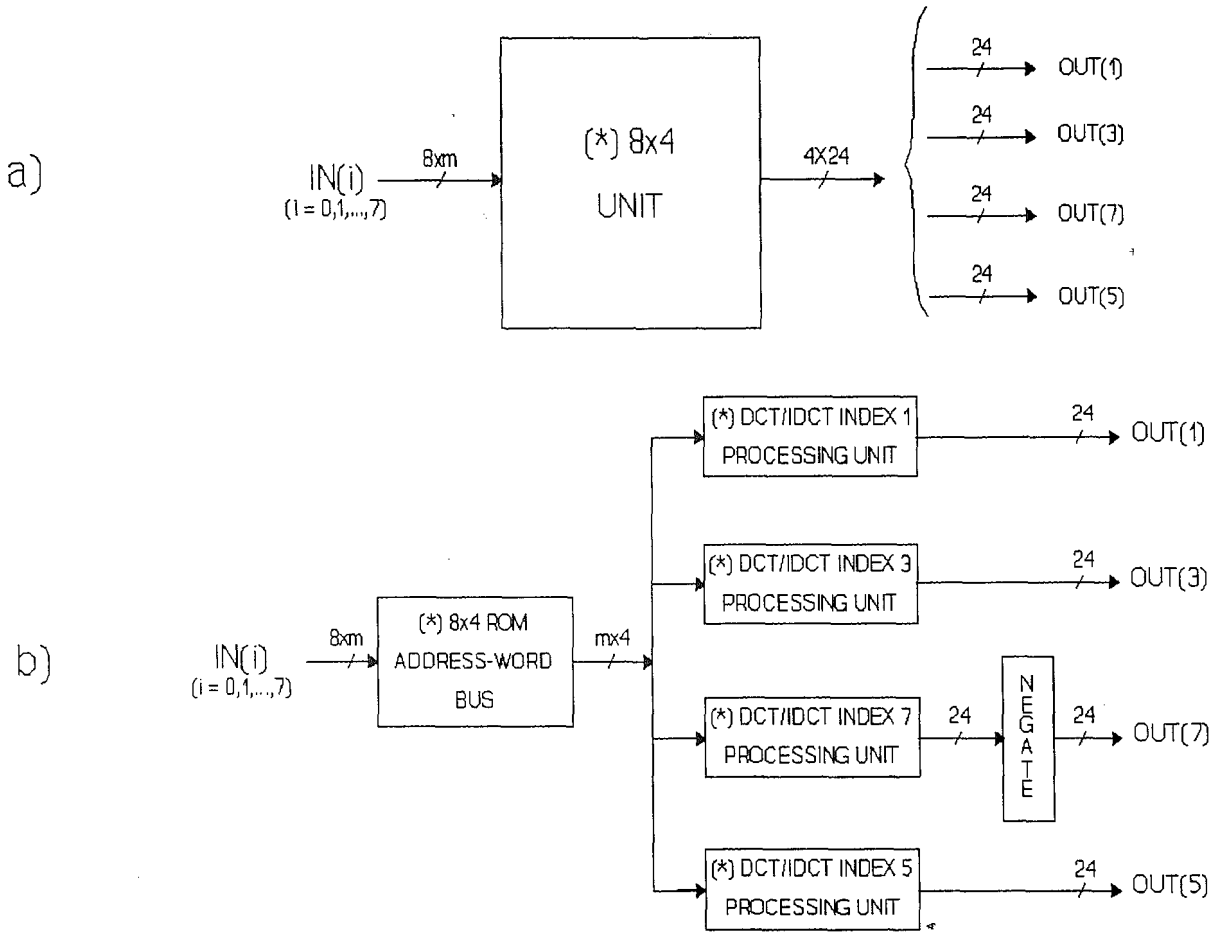

Figure A9. Symbol representation (a) and block diagram (b) of the 8x4 (*8x4) Unit. Where m = 16 for the 8x4 Unit, and m = 24 for the *8x4 Unit

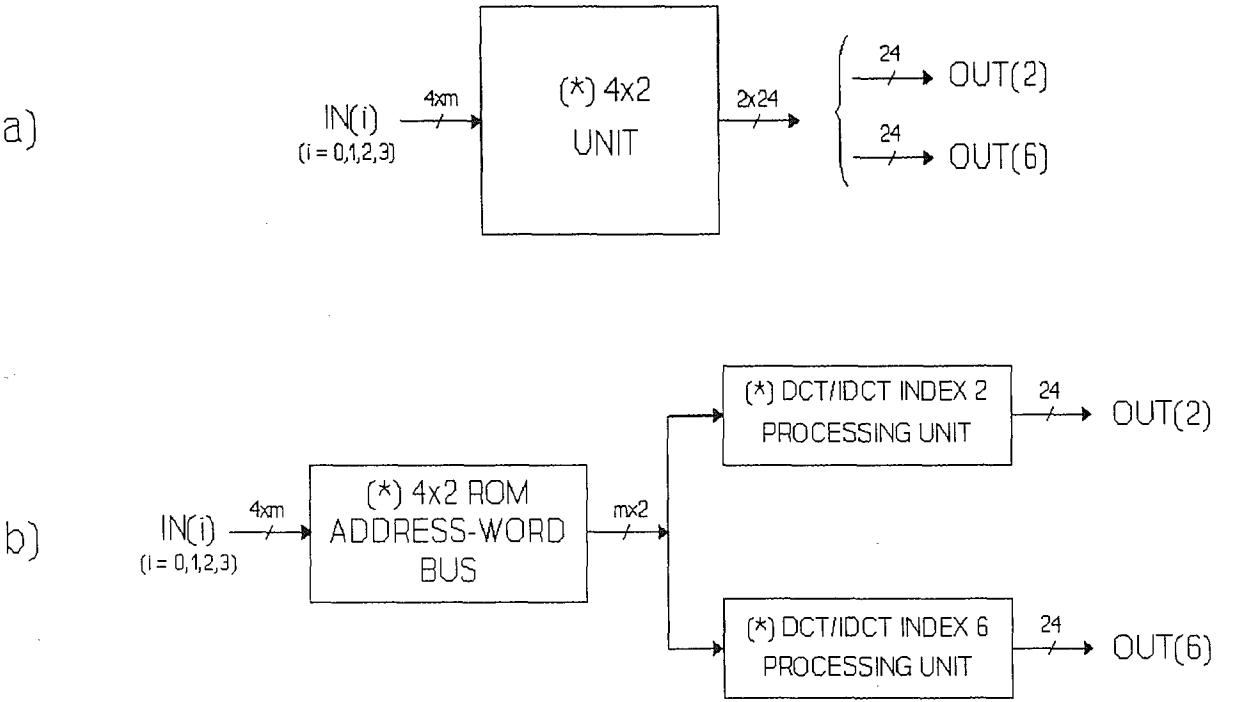

Figure A10. Symbol representation (a) and block diagram (b) of the 4x2 (*4x2) Unit. Where m = 16 for the 4x2 Unit, and m = 24 for the *4x2 Unit.

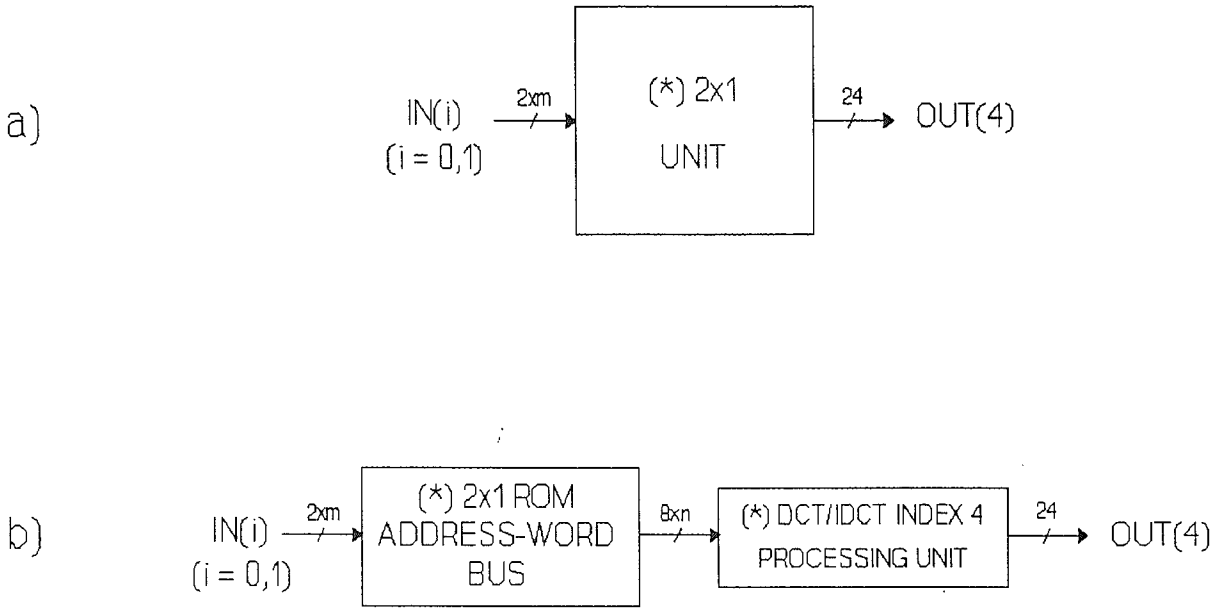

Figure A11. Symbol representation (a) and block diagram (b) of the 2x1 (*2x1) Unit. Where m = 16 & n = 2 for the 2x1 Unit, and m = 24 & n = 3 for the *2x1 Unit.

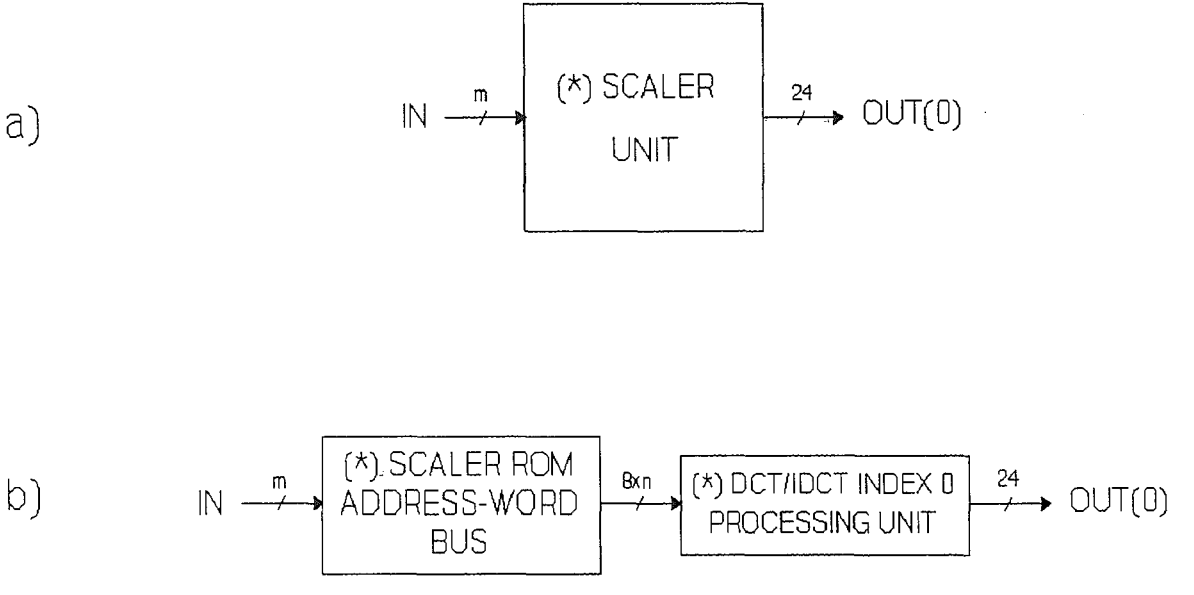

Figure A12. Symbol representation (a) and block diagram (b) of the scaler (*scaler) Unit. Where m = 16 & n = 2 for the Scaler Unit, and m = 24 & n = 3 for the *Scaler Unit.

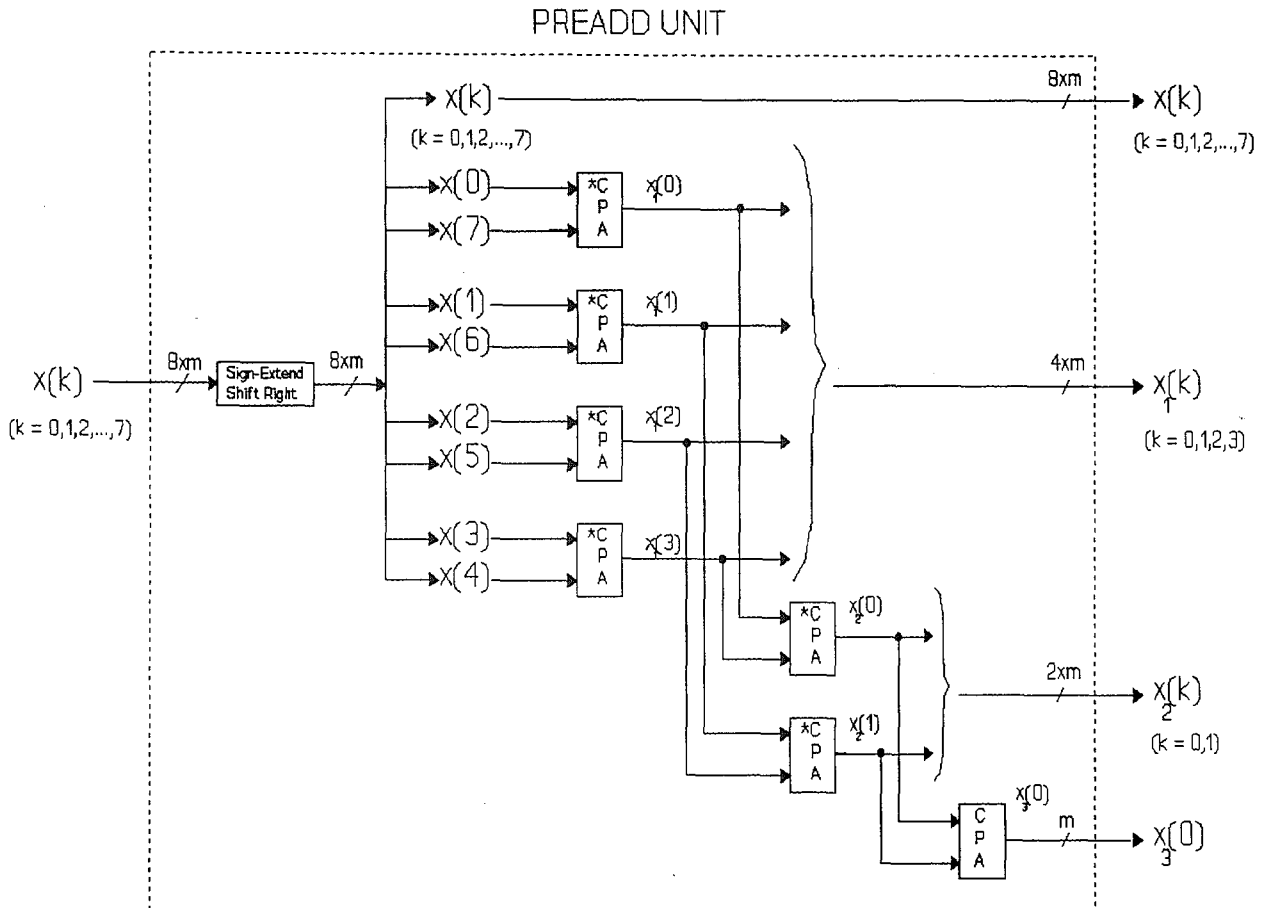

Figure A13. Preadd Unit hardware design. Note: *CPA represents an output scaling CPA (i.e, divide by two).

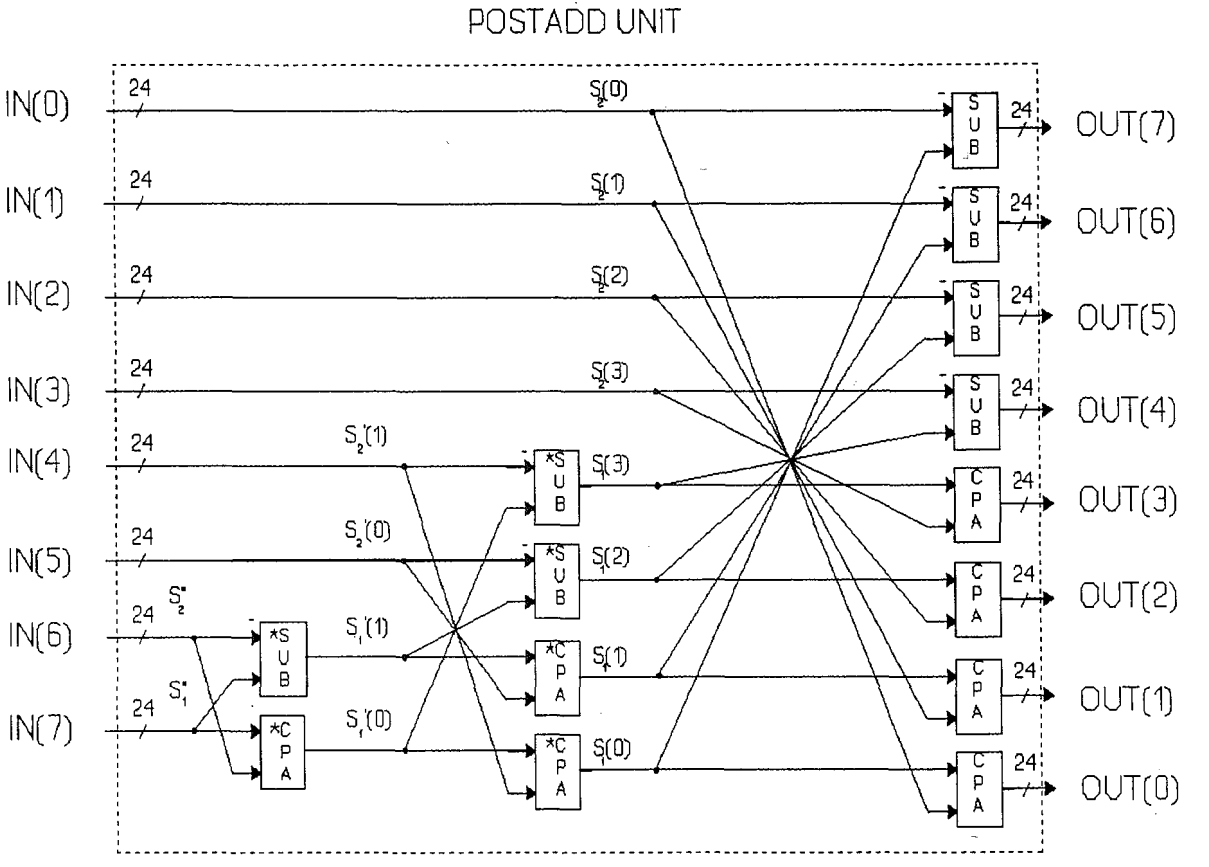

Figure A14. Postadd Unit hardware design. Note: *CPA and *SUB represent output scaling components.

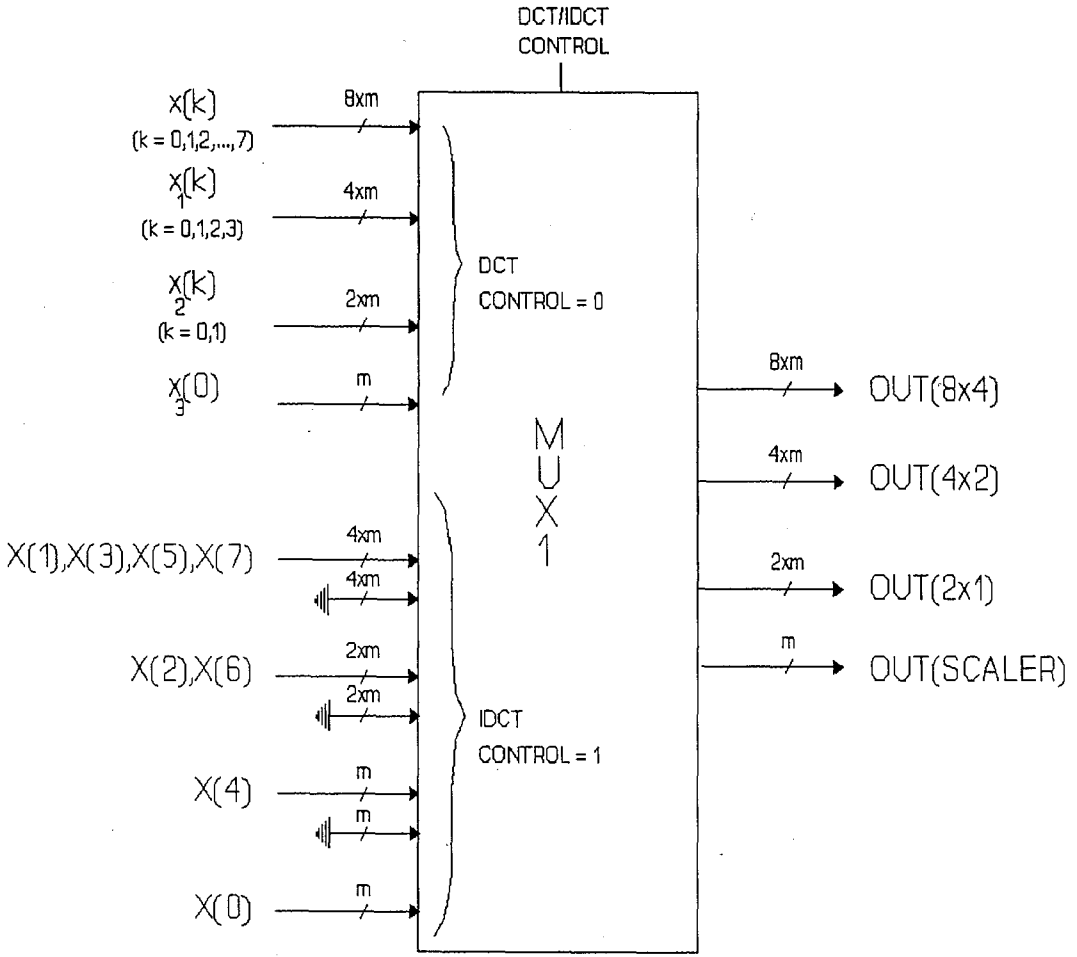

Figure A15. Multiplexer 1.

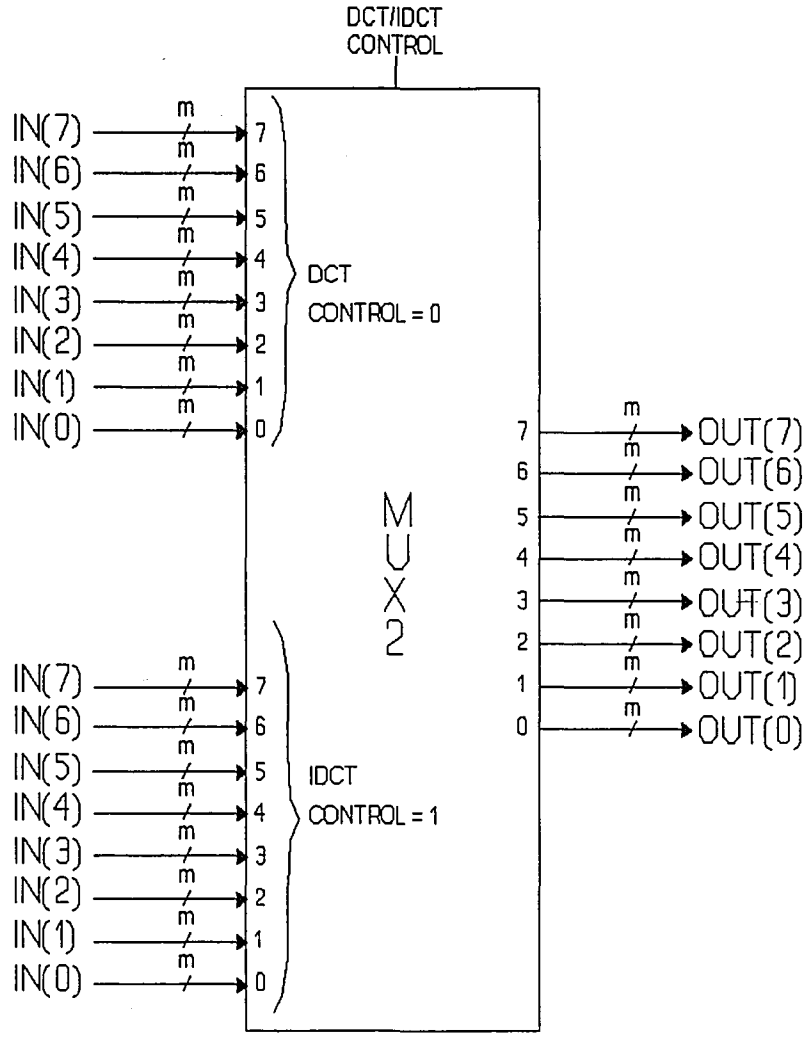

Figure A16. Multiplexer 2.

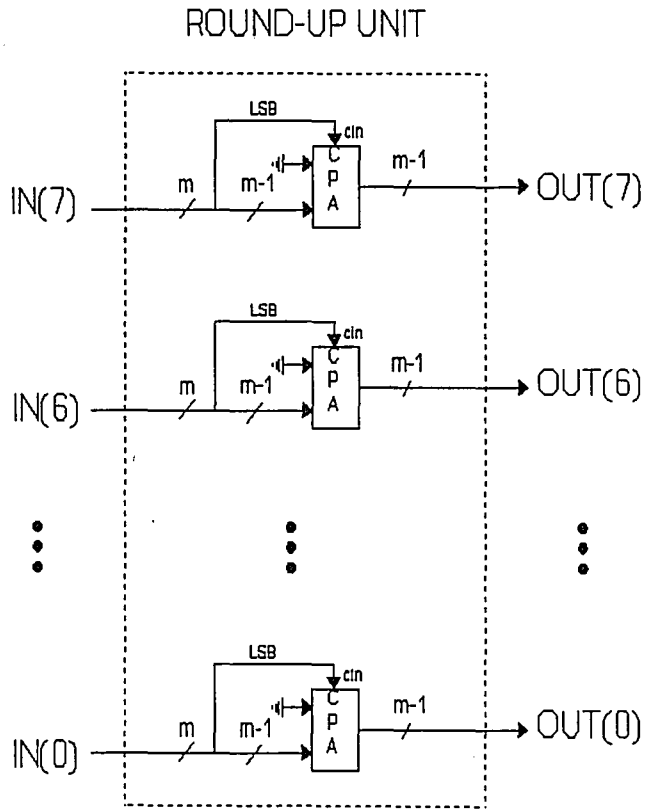

Figure A17. Round-up Unit hardware design.

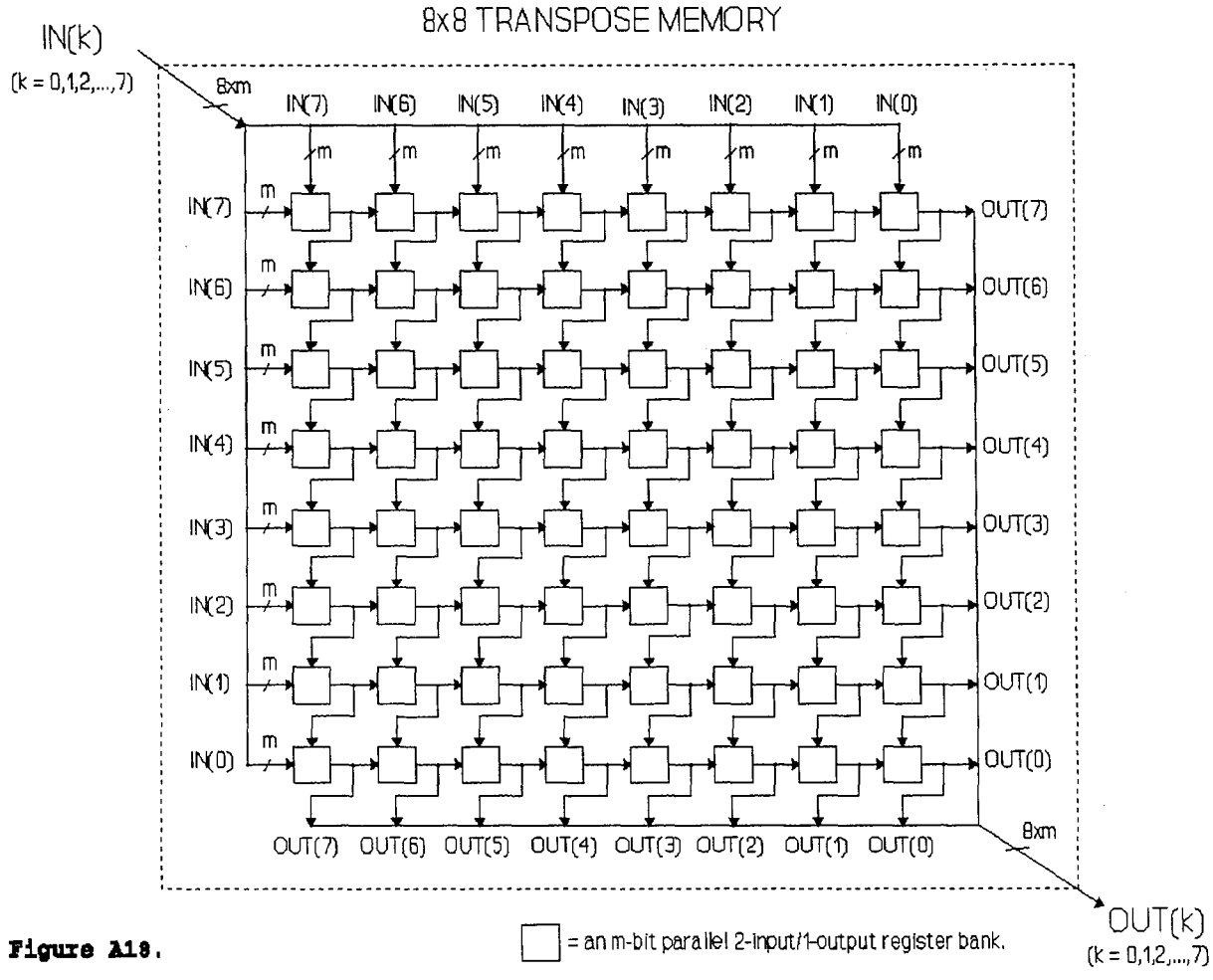
8x8 TRANSPOSE MEMORY
IN(k)
(k = 0,1,2,...,7)
8xm
IN(7) IN(6) IN(5) IN(4) IN(3) IN(2) IN(1) IN(0)
m
OUT(7)
OUT(6)
OUT(5)
OUT(4)
OUT(3)
OUT(2)
OUT(1)
OUT(0)
OUT(k)
(k = 0,1,2,...,7)
= an m-bit parallel 2-input/1-output register bank.

Figure A18.

APPENDIX B: ROM CONTENTS

There are two kinds of DCT/IDCT Index "k" Processing Units presented. One implements biased-redundant-binary adders (BRBA) (Figure A1), used in the 8x4 and 4x2 (*8x4 and *4x2) Units, and the other implements carry-save adders (Figure A2), used in the 2x1 and Scaler (*2x1 and *Scaler) Units. Each processing unit uses ROMs of different memory sizes and content. Design #1 and #2 2-D DCT/IDCT Processors depend on the ROMs having the correct content in order to successfully meet the IEEE standard specifications. Since this a critical part of the design, the ROM contents of the designs are given below.

The variable k represents the DCT/IDCT coefficient being calculated, where k = {0,1,...,7}. As mentioned, the 8x4 (k = {1,3,7,5}) Unit and the 4x2 (k = {2,6}) Unit use the BRBA configuration. Remember that an (*) denotes the 24 partial product generations and additions used in Design #2 2-D DCT/IDCT Processor. The corresponding index "k" of the 8x4 Unit and of the *8x4 Unit use identical ROMs. The same applies for the 4x2 and *4x2 Units.

8x4 and *8x4 Unit ROM Contents

Figure A4 shows the 8x4 and *8x4 ROM Address-Word Bus design. The ROM words generated are of the form $b_3b_2b_1b_0$. There are four ROMs in which the contents are determined from the skew-circular convolution matrix (Chapter 3, Equation (15) and (18)). Table B1 shows the 24-bit

hexadecimal contents of the four ROMs.

Computations

$$\text{ROM Index 1} = \left[b_3 \cdot C(0) + b_2 \cdot C(1) + b_1 \cdot C(2) + b_0 \cdot C(3) \right] \div 4$$

$$\text{ROM Index 3} = \left[b_3 \cdot C(1) + b_2 \cdot C(2) + b_1 \cdot C(3) - b_0 \cdot C(0) \right] \div 4$$

$$\text{ROM Index 7} = \left[b_3 \cdot C(2) + b_2 \cdot C(3) - b_1 \cdot C(0) - b_0 \cdot C(1) \right] \div 4$$

$$\text{ROM Index 5} = \left[b_3 \cdot C(3) - b_2 \cdot C(0) - b_1 \cdot C(1) - b_0 \cdot C(2) \right] \div 4$$

C(m) is defined in Chapter 3, Equation (18). These contents are scaled by four which has to do with the dynamic range issue mentioned in Chapter 5.

ROM Address-Word $b_3b_2b_1b_0$	ROM Index 1	ROM Index 3	ROM Index 7	ROM Index 5
0000	000000	000000	000000	000000
0001	08e39e	f04eb4	f2b24d	031f17
0010	fce0e9	08e39e	f04eb4	f2b24d
0011	05c487	f93252	e30101	f5d164
0100	0d4db3	fce0e9	08e39e	f04eb4
0101	163151	ed2f9d	fb95eb	f36dcb
0110	0a2e9c	05c487	f93252	e30101
0111	13123a	f6133b	ebe49f	e62018
1000	0fb14c	0d4db3	fce0e9	08e39e
1001	1894ea	fd9c67	ef9336	0c02b5
1010	0c9235	163151	ed2f9d	fb95eb
1011	1575d2	068005	dfe1ea	feb502
1100	1cfeff	0a2e9c	05c487	f93252
1101	25e29d	fa7d50	f876d4	fc5169
1110	19dfe8	13123a	f6133b	ebe49f
1111	22c386	0360ee	e8c588	ef03b6

Table B1. ROM contents (hexadecimal #) for the 8x4 and *8x4 Units.

4x2 and *4x2 Unit ROM Contents

Figure A5 shows the 4x2 and *4x2 ROM Address-Word Bus design which generates address words of the form b_1b_0. There are two ROMs which are determined from the skew-circular convolution matrix defined in Chapter 3, Equation (21) and (22). Table B2 shows the 24-bit hexadecimal contents of the two ROMs.

Computations

ROM Index 2 = $\left[b_1 \bullet C_1(0) + b_0 \bullet C_1(1) \right] \div 2$

ROM Index 6 = $\left[b_1 \bullet C_1(1) - b_0 \bullet C_1(0) \right] \div 2$

$C_1(m)$ is defined in Chapter 3, Equation (22). These ROMs are scaled by two.

ROM Address-Word b_1b_0	ROM Index 2	ROM Index 6
00	000000	000000
01	0c3ef1	e26f94
10	1d906c	0c3ef1
11	29cf5d	eeae86

Table B2. ROM contents (hexadecimal #) for the 4x2 and *4x2 Units.

2x1 and Scaler Unit Rom Contents

These units implement the carry save adder configuration shown in Figure A2. As Equation (31) (Chapter 3) shows, the constants C_2 and C_3 are identical. Thus, the ROM contents of these two units are the same. The ROMs of this design employ a control bit (denoted by control). Figure A5 shows the 2x1 ROM Address-Word Bus design,

whereas, Figure A8.b shows the Scaler ROM Address-Word Bus design. Both generate address words of the form b_1b_0. Thus, the control bit and the address word can be thought of as an entire address word, control b_1b_0. Table B3 shows the 24-bit hexadecimal contents of the two ROMs.

Computation

ROM Index 0 (and 4) = $(-1)^{control} \bullet b_1 \bullet C_2 + b_0 \bullet C_2/2$

C_2 is defined in Chapter 3, Equation (31). Unlike the BRBA ROMs, these ROM units are not scaled.

ROM Address-Word control b_1b_0	ROM Index 0 (& 4)
000	000000
001	16a09e
010	2d413d
011	43e1db
100	000000
101	16a09e
110	d2bec3
111	e95f62

Table B3. ROM contents (hexadecimal #) for the 2x1 and Scaler Units.

*2x1 and *Scaler Unit ROM Contents

These units implement the 24 partial product generations and additions mentioned previously. Whereas the BRBA configuration uses an extra ROM and CPA per stage (Figure A1.d) to handle the 24 partial product case (denoted by *), the CSA configuration sections the ROM address words into eight groups of three bits ($b_2b_1b_0$). Thus, instead of

modifying the hardware design, only the ROMs replaced. Table B4 shows the 24-bit hexadecimal contents for the special ROMs (*ROM).

Computation

*ROM Index 0 (& 4) = $(-1)^{control} \bullet b_2 \bullet C_2 + b_1 \bullet C_2/2 + b_0 \bullet C_2/4$

*ROM Address-Word control $b_2b_1b_0$	*ROM Index 0 (& 4)
0000	000000
0001	0b504f
0010	16a09e
0011	21f0ee
0100	2d413d
0101	38918c
0110	43e1db
0111	4f322a
1000	000000
1001	0b504f
1010	16a09e
1011	21f0ee
1100	d2bec3
1101	de0f12
1110	e95f62
1111	f4afb1

Table B4. ROM contents (hexadecimal #) for the *2x1 and *Scaler Units.

For all of the ROMs discussed above, the 25-bit content was determined first. Then, this was rounded-up based on the LSB to give the 24-bit result.

BIOGRAPHY

Maurice D. Caldwell received his BSEE from Lafayette College in 1989 where he was a member of the varsity football team. He received his MSEE from Lehigh University in 1992 where he was a research assistant, a member of the National Society of Black Engineers, and a tutor for local elementary school students. His research interests include digital signal processing, VLSI design and image processing.

END OF TITLE